UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

19|20

Trabajo fin de grado

Sumarización automática de noticias en español

Inés Fernández Campos

Campus Internacional excelencia UAM CSIC+

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Sumarización automática de noticias en español

Autor: Inés Fernández Campos
Tutor: Manuel Sánchez-Montañés

junio 2020

Inés Fernández Campos
Sumarización automática de noticias en español

Inés Fernández Campos

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Con paciencia y una caña se escapó Chucho la araña

Refranero español

Agradecimientos

En primer lugar quiero agradecer a mis padres su apoyo durante mis estudios en la Escuela y durante la realización de este proyecto en esta situación tan peculiar que nos ha tocado vivir.

Agradecerle, también, su trabajo y esfuerzo a mi tutor, Manuel Montañés, que me ha ayudado y enseñado a lo largo del desarrollo del proyecto.

Por último, a mi pareja, y a mi gentecilla, por su paciencia, su motivación y sus infinitos ánimos, y a mis compañeros y amigos, con los que he compartido clases y prácticas, y que han trabajado, sufrido y celebrado conmigo durante estos años.

Resumen

Con la ingente cantidad de información que se genera diariamente y a la cual tenemos acceso gracias a Internet, la sumarización de textos ha resultado ser una herramienta increíblemente útil, no solo como un medio para ganar eficiencia en su lectura sino también para despojar estos textos de la información irrelevante o secundaria que puedan contener.

Si bien algunos textos traen consigo algún tipo de resumen o abstracto, como pueda tener por ejemplo este trabajo, los artículos periodísticos no, es por eso que en este trabajo se estudian distintas técnicas de analítica de texto para sumarizar noticias en español. El trabajo se desarrolla en Python y tiene como finalidad la obtención de resúmenes extractivos y la generación de los títulos de noticias extraídas de publicaciones nacionales.

Tras el estudio del estado actual del estado del arte, decidimos implementar dos técnicas de sumarización extractiva para poder comparar sus resultados y construir un modelo sequence-to-sequence que permita la generación de los titulares de las noticias.

Los resultados obtenidos demuestran la capacidad para llevar a cabo la tarea así como revelan algunos problemas, como son el hacer uso de frases que hacen referencia a información anterior que no han sido seleccionadas o la complicación del resumen de entrevistas.

Palabras clave

Sumarización automática, aprendizaje profundo, generación de títulos, resumen extractivo

Abstract

Given the amount of information that is generated on a daily basis and that can be found in the Internet, automatic summarization has been proven to be an incredibly useful tool, not only as a way to be more efficient in its reading and consumption but also to rid the texts of all irrelevant information they might contain.

Knowing that some texts include a summary or abstract in their structures, as do scholarly papers, we chose to focus our work on the study of different text analytic techniques applied to Spanish news articles. Our end goal was to produce extractive summaries as well as generate the headlines of some texts taken from national publications.
After a thorough study of the works that had already been done on the field, we decided to implement two extractive summarization techniques in order to analyze the results and build a sequence to sequence model that would allow us to generate the news titles.

The results we obtained were promising and the summaries truthful. However, they do reveal a couple of problems as the summaries tend to reference information that does not appear in the summaries or show complications depending on the type of the text given.

Keywords

Automatic text summarization, deep learning, headline generation, extractive summaries

Índice

Listas

Lista de algoritmos

Lista de ecuaciones

Lista de figuras

1

Introducción

A principios del siglo XX da comienzo lo que se conoce como la Era de la Información, un período temporal que se caracteriza por un cambio de enfoque en la industria que pasa a depender fuertemente de la industria tradicional, heredada de la revolución industrial, a una industria que se basa en las Tecnologías de la Información y la Comunicación (TIC) . Con este cambio en el foco de atención principal, es evidente que la producción de datos no ha hecho más que aumentar desde el comienzo de esta era y resultaría lógico pensar que esta tendencia no va a cambiar.

De este auge de información y documentos surge la necesidad de adquirir eficientemente y con precisión la esencia de estos. En el diccionario de la Real Academia Española se define el verbo resumir de la siguiente manera:

> Reducir a términos breves y precisos, o considerar tan solo y repetir abreviadamente lo esencial de un asunto o materia.

Así pues, el resumen se convierte en uno de los formatos más convenientes de tratar la gran cantidad de información de la que nos vemos rodeados.

De la mano al desarrollo de las TIC, la globalización y esfuerzo por una conexión total, nos encontramos a que frecuentemente la documentación e información se redacta en inglés. Así la mayoría de trabajos sobre la sumarización automática se encuentran y desarrollan en inglés.

Con este trabajo se pretende llevar la sumarización al castellano y es por ello que todos los artículos vienen de noticias publicadas en periódicos nacionales. Esto sin duda ha cambiado nuestra idea de proyecto inicial, la generación de resúmenes abstractivos, por la generación de títulos de artículos en castellano y el resumen extractivo de estos. El proyecto consta por tanto de dos partes, la segunda formando parte del ámbito del Aprendizaje Profundo (Deep Learning).

1.1. Objetivos

El principal objetivo que persigue este Trabajo de Fin de Grado (TFG) es la obtención de resúmenes a través del aprendizaje automático, la generación automática de titulares mediante el uso de Deep Learning y la evaluación de los resultados obtenidos.

Para ello se marcan los siguientes objetivos específicos:

- Estudio de las técnicas de sumarización existentes.
- Estudio de las métricas de evaluación para la sumarización de texto.
- Implementación de técnicas para la generación de resúmenes.
- Construcción de un modelo para la generación de titulares de textos.
- Evaluación de los resultados obtenidos.

1.2. Estructura del documento

Una vez hecha la introducción y presentados los objetivos, en el capítulo 2 se realiza un estudio del estado del arte y se presenta la documentación en la que nos hemos basado para el proyecto.

En el capítulo 3, abordamos la metodología aplicada para la obtención de los objetivos expuestos. Tratamos los métodos implementados para la obtención de sumarios extractivos y se expone el modelo construido para abordar la tarea de la generación de títulos, además de los métodos de evaluación de ambas tareas.

El capítulo 4 se centra en el análisis de los resultados obtenidos basándose en la metodología expuesta en el capítulo anterior. Y finalmente, el capítulo 5 trata las conclusiones y el trabajo futuro.

2

Estado del arte

En este capítulo situaremos el proyecto llevado a cabo dentro del campo del procesamiento del lenguaje natural. Se presentarán, además, algunos de los trabajos relevantes y que hayan sido de importancia para nuestro proyecto, así como los algoritmos y soluciones existentes relevantes.

2.1. Procesamiento del Lenguaje Natural

El proyecto llevado a cabo se enmarca dentro del campo del procesamiento del lenguaje natural (NLP, Natural Language Processing en inglés), de la inteligencia artificial. Existen un gran número de obras de amplio interés introductorias al campo, como puedan ser la de P.M.Nadkarni [1] o el estudio sobre la aplicación de NLP a la recuperación de información (IR,Information Retrieval) de T.E.Doszkocs [2].

El Procesamiento del Lenguaje Natural estudia la comunicación entre máquinas y personas a través de lenguas naturales como puedan ser el español o el inglés. Dada la variedad de formas en las que se puede dar una lengua natural, por escrito, en forma de texto, de manera oral, usando la voz, e incluso mediante signos, el Procesamiento del Lenguaje Natural, se puede usar para resolver muchos tipos de problemas, como puedan ser el reconocimiento de voz, la traducción entre idiomas o, en nuestro caso, el resumen de textos [1].

Esta disciplina surge de la convergencia entre la inteligencia artificial y la lingüística. Mientras que la lingüística es el estudio del lenguaje, su gramática, semántica y fonética a partir del uso de reglas que permiten definirlo [3], la inteligencia artificial se puede definir como la rama que persigue automatizar tareas intelectuales normalmente llevadas a cabo por humanos [4].

Dentro del campo del Procesamiento del Lenguaje Natural, se pueden diferenciar dos aproximaciones principales: procesamiento estadístico del lenguaje natural, y el procesamiento lingüístico del lenguaje natural, en inglés conocido como rule-based Natural Language Processing [5]. El procesamiento estadístico del lenguaje natural es el modelo por excelencia de los sistemas de extracción de información, Information Retrieval (IR) , donde cada documento se describe por palabras clave, en este caso denominadas términos índice. Este modelo se basa en el "bag of words"("bolsa de palabras")

donde cada palabra o término tiene un peso, generalmente dado a partir del número de veces que este aparece, que caracteriza su importancia [6].

Por otro lado, el procesamiento lingüístico del lenguaje natural, busca realmente entender el lenguaje en todos sus niveles, morfológico, semántico, sintáctico y pragmático [6]. Para ello, debe considerar el orden, estructura y significado de las palabras que lo componen, cosa que el estadístico, por sí solo, no hace. Este enfoque más centrado en el lenguaje en sí y su estructura nace en los años 50 con Chomsky.

Aunque estas dos aproximaciones persigan diferentes enfoques, en muchas ocasiones los sistemas NLP utilizan una combinación de técnicas de ambas. J.Nivre muestra como muchos de los métodos estadísticos muchas veces emplean técnicas del campo de la lingüística y cómo también en métodos lingüísticos es muy común que haya presencia de métodos estadísticos [5].

2.2. Sumarización de textos

Si bien ahora sabemos qué es el Natural Language Processing (NLP) , nos queda abordar el porqué es relevante para nosotros este campo de estudio. Nuestro proyecto consiste en la obtención automática de resúmenes y títulos a partir de noticias en castellano. Es decir, en este TFG llevamos a cabo la sumarización de esas noticias en su resumen y titular. La sumarización de textos es un problema perteneciente al Procesamiento del Lenguaje Natural ya que se trata de reducir el tamaño de un texto dado para producir otro cuya idea principal sea similar.

Resumir o sumarizar un texto tiene numerosas ventajas, como por ejemplo, que leerlo requiera menos tiempo, que nos permite evaluar cómo de bien se ajusta un documento a lo que buscamos sin necesidad de leerlo entero (como hacen los resúmenes de los artículos científicos) o, si se trata de un sistema automático de sumarización (ATS), también tiene la ventaja de ser más objetivo que un agente sumarizador humano [7].

Los resúmenes se pueden clasificar de distintas maneras según el tipo de entrada que reciben, su objetivo y la salida que generan. Según la entrada que pueda recibir, hablamos de sumarizadores multidocumento, si la entrada es más larga y ocupa varias páginas, o monodocumento, donde la entrada suele ser más corta [8].

Por objetivo, encontramos tres tipos de resúmenes. Los resúmenes generales son aquellos donde la información se trata de manera homogénea y no se asume un ámbito específico. El segundo tipo lo conforman los resúmenes que son específicos de un campo y aprovechan el conocimiento sobre este campo para crear un sumario más preciso. Por último encontramos aquellos ambientados a que la información que contiene el resumen sirva para contestar preguntas del lenguaje natural, preguntas que le puedes hacer por ejemplo a un bot online.

Por último, según el tipo de salida que generen, los sumarizadores pueden ser abstractivos o extractivos. Los abstractivos son aquellos que generan sus propias frases para dar información más completa, como los que haría una persona. Los extractivos son sumarizadores que forman su resumen a partir de las frases, o trozos de frases, más significativos del texto [9].

En el proyecto, la obtención de resúmenes y la generación de titulares la hemos realizado de maneras diferentes. Para la generación de los títulos de las noticias hemos usado deep learning, o aprendizaje profundo, del cuál hablamos más adelante en este capítulo. Mientras que para la obtención de los resúmenes hemos implementado distintos métodos que no entran dentro del campo del aprendizaje profundo.

El hecho de que hayamos usado distintas técnicas y algoritmos para esas tareas se debe a que la primera, la generación de títulos, se enmarca dentro de la sumarización abstractiva, donde necesitamos, para aúnar el total de una noticia que nuestro ATS conozca el contexto de esta y la relación de las palabras que la componen. Para la segunda, la obtención de los resúmenes, nos introducimos en el campo de la sumarización extractiva.

2.2.1. Sumarización extractiva

Dentro del Machine Learning encontramos tres grandes categorías a las que pueden pertenecer los problemas, aprendizaje supervisado y aprendizaje no supervisado y aprendizaje por refuerzo [4]. En este proyecto, nos interesan las dos primeras.

En el aprendizaje supervisado contamos con un modelo que aprende basándose en un dataset que contiene tanto los datos de entrada como los datos esperados de salida, los targets que se esperan generar. En el aprendizaje no supervisado no se cuenta con un dataset con unas salidas de las que aprender y por tanto, se busca encontrar patrones en los datos de entrada.

En esta subsección procedemos a explicar los métodos de sumarización extractiva más relevantes para nuestro proyecto, aunque los métodos de los que vamos a hablar pertenecen a la segunda categoría mencionada en el párrafo anterior. Muchas de estas técnicas han sido recopiladas en el trabajo de Allahyari y colaboradores [10].

Uno de los primeros métodos que surge es creado por Luhn en 1950 [11]. Su objetivo era generar sumarios a partir de textos técnicos de manera automática. Para ello, Luhn establece una medida de importancia para cada palabra y frase, las frases con la puntuación más alta son las que conforman finalmente el sumario [11]. Esta medida de importancia es la frecuencia con la que aparece cada termino y para evitar que ciertas palabras que tendemos a repetir más tengan una mayor importancia, establece un rango de frecuencia con el que trabaja. Esas palabras que eliminaba Luhn son lo que ahora conocemos como "stop-words". Por ejemplo, algunas stop-words típicas en español son ’a’, ’de’, ’y’.

Posteriormente, Vanderwenden y colaboradores [12] proponen un nuevo método conocido bajo el nombre de SumBasic. La idea detrás de este algoritmo es que las palabras que más aparecen en el documento tienen una probabilidad mayor de aparecer en un resumen realizado por un agente humano que aquellas que aparecen menos [13]. A diferencia del método de Luhn, SumBasic recalcula la probabilidad de que una palabra aparezca después de haber seleccionado la frase con mayor puntuación general lo que permite que otras palabras con una probabilidad inicial menor obtengan un mayor peso.

Otra manera de contabilizar la importancia de las palabras que aparecen en un texto sin necesidad de excluir las stop-words, como hacía Luhn, es usando algoritmos basados en Term Frequency Inverse Document Frequency (TF-IDF) . Esta técnica mide la importancia de las palabras en un texto identificando las palabras que sean muy comunes [10].

Para el cálculo de los pesos deberemos tener dos partes, una que haga referencia a la frecuencia con la que aparecen las palabras (TF), y una segunda que haga alusión al factor de inversión (IDF) [14]. En la ecuación 2.1 propuesta por Allahyari y colaboradores [10], el primer cociente es la frecuencia de una palabra (w) en un documento (n), el segundo cociente cumple la función de factor inversor, teniendo en cuenta las veces que la palabra aparece en la totalidad de los documentos (N).

$$peso_w = f_n(w) * \log \frac{|N|}{f_N(w)} \tag{2.1}$$

Hay muchas otras técnicas que no se basan principalmente en la frecuencia de sus términos sino que se basan en grafos o métodos algebraicos para sumarizar. Uno de los algoritmos más conocidos basado en grafos es TextRank, inspirado en PageRank, una de las técnicas que usa Google para determinar la relevancia de una página a través de las páginas que la referencian [15]. Para ello, se presentan las páginas como si fueran nodos de un grafo, conectando una página web con sus referencias, y se calcula la probabilidad de que un usuario estando en una página W_i llegue a una de sus referencias W_j. Estas probabilidades son los pesos entre nodos. Se presenta más abajo un ejemplo con cuatro páginas webs 2.1(a) y se procede al cálculo de probabilidades en la matriz 2.1(b).

página	referencias
Web1	[Web2, Web4]
Web2	[Web4]
Web3	[ ]
Web4	[Web1,Web3]

(a) Esquema de webs y sus referencias

	Web1	Web2	Web3	Web4
Web1	0	0.5	0	0.5
Web2	0	0	0	1.0
Web3	0.25	0.25	0.25	0.25
Web4	0.5	0	0.5	0

(b) Matriz de probabilidades o puntuaciones para las Webs1-4

Tabla 2.1: Explicación del algoritmo PageRank

Teniendo la idea básica del funcionamiento de PageRank, será mucho más sencillo explicar su aplicación a los sumarios con TextRank. En este algoritmo, las páginas webs son sustituidas por las frases del texto original a resumir, y se crean matrices de similitud entre cada una de sus frases. Si

visualizamos esto en forma de grafo, las frases en las que se ha dividido el documento son los nodos del grafo, y sus aristas, la similitud entre cada nodo. Los nodos con más conexiones, mayor similitud, tienen muchas posibilidades de formar parte del resumen final. Esta técnica de sumarización se usa tanto para sumarización extractiva a nivel de frases como de palabras clave dando resultados que compiten e incluso superan algunas técnicas de extracción de aprendizaje supervisado [16].

2.2.2. Sumarización abstractiva

Los trabajos y métodos que hemos cubierto en la subsección anterior son de gran peso y han dado buenos resultados en el campo de la sumarización extractiva. Sin embargo, para la generación de titulares, las técnicas vistas no serán adecuadas ya que requeriremos del contexto completo de las noticias y no valdrá con simplemente extraer la frase más puntuada del texto.
Por ello, debemos estudiar también la rama abstractiva de la sumarización y los métodos comúnmente empleados.

La tarea de la abstracción es una tarea compleja y los métodos empleados usan aprendizaje automático, generalmente, aprendizaje supervisado. En esta subsección no pretendemos cubrir el deep learning, tema en el que nos adentraremos en la siguiente sección, simplemente compartir algunas de las técnicas que han ido dando mejores resultados para la generación de tanto de titulares como de sumarios cortos.

Word Embeddings

En comparación con la sumarización extractiva, en los sumarios abstractivos, gran parte de la complejidad surge de la necesidad de comprender el contexto general del texto original y la relación existente entre palabras. Hay técnicas de codificación de textos muy simples, como one hot encoding, donde las palabras son traducidas a vectores de dimensiones (1, vocab_size), siendo vocab_size el tamaño del vocabulario del texto, y dejando todos los índices del vector a 0 exceptuando el índice de la palabra correspondiente cuyo vector se esté creando presentan grandes problemas como la dimensión de los vectores creados, y la falta de relación entre las palabras que representan.

Una técnica habitual para lidiar con esta complicación es el uso de word embeddings. Los word embeddings son representaciones vectoriales de palabras capaces de aunar su contexto en un documento así como su similitud semántica y sintáctica y la relación que mantiene con otras palabras de este [17]. Así, las técnicas de embeddings que vamos a tratar aquí, demuestran ser una técnica muy potente y, usándolos correctamente, dos palabras con un significado cercano que aparezcan en contextos similares estarán en espacios vectoriales cercanos.

Como ejemplo, en la figura 2.1 hemos usado un proyector de embeddings [18] para representar algunas de las palabras más cercanas a la palabra pandemia.

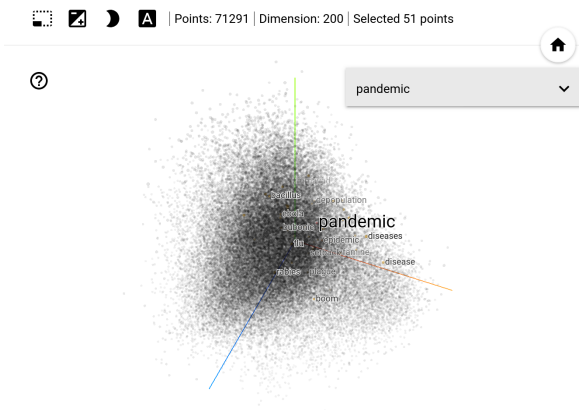


Figura 2.1: Representación word2vec de la palabra pandemic [18]

Dentro de los word embeddings, existen dos técnicas predominantes de construirlos: Skip-gram, Continuous Bag Of Words (CBOW) , métodos implementados por Word2Vec, y embeddings Global Vectors for Word Representation (GloVe) .

Los métodos que se pueden generar a partir de Word2Vec, Skip-gram y CBOW, son dos arquitecturas que se usan para aprender la representación vectorial de las palabras a través de redes neuronales [19]. La figura 2.2 muestra de manera gráfica ambas arquitecturas.

La arquitectura CBOW utiliza las palabras vecinas para predecir la palabra del medio que sería el target, mientras que Skip-gram utiliza una palabra para predecir la probabilidad de que se haye dentro de un contexto [20]. Ambos métodos utilizan una ventana, en el caso de CBOW, la ventana delimita el número de palabras que rodean al target que se usan como input, y en el caso de Skip-gram, la ventana representa el número de palabras contexto que debe predecir.

A la hora de elegir entre ambos embeddings, según Mikolov, CBOW es más rápido y ofrece representaciones mejores para palabras comunes. Sin embargo, ante la presencia de palabras poco comunes o menos datos en los que basarse, Skip-gram da mejores resultados. Existen numerosas obras que tratan con más detalle estas arquitecturas, merecen un vistazo sin duda los trabajos de Mikolov et al. 2003 [20], Rumelhart et al. [21] y Bengio et al. [22].

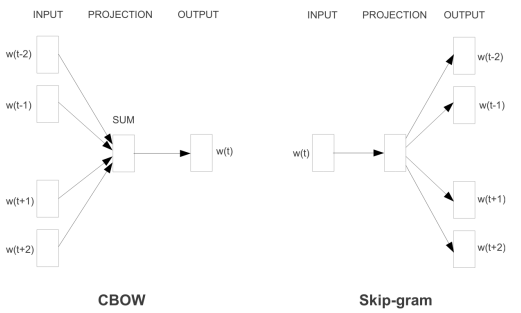


Figura 2.2: Del artículo: Exploiting similarities among languages for machine translation [20]

Por otro lado, los embeddings GloVe presentan la ventaja frente a los embeddings Word2Vec de utilizar no solo el contexto local de las palabras cuyos vectores se quiere obtener (la ventana de la que hablabamos) sino de incorporar además información global (GloVe viene de Global Vectors) a través de matrices de co-ocurrencia [23]. Estas matrices dan información contextual indicando cuantas veces aparece una palabra en el contexto de otra.

El modelo empleado se explica detalladamente en el trabajo de J.Pennington et al. [24] en el que se hace una comparativa de los vectores resultantes obtenidos con GloVe frente a las técnicas que ya hemos visto.

Generación de Lenguaje Natural

Otro campo que nos urge abordar relacionado con la generación de titulares a través de métodos de sumarización abstractiva, es el campo de la Generación de Lenguaje Natural (GLN) . La GLN pertenece al campo de la inteligencia artificial, más concretamente, E. Reiter y R. Dale lo definen como:

> El subcampo de la inteligencia artificial y lingüística computacional que trata de la construcción de sistemas que tienen como objetivo la producción de texto comprensible en inglés u otros idiomas a partir de representaciones no lingüísticas de la información [25].

Esta definición proviene de su trabajo “Building Applied Natural Language Generation Systems“ en el cual hacen un resumen aplicado del GLN que sin duda vale la pena.

También es amplia la literatura donde se tratan los sistemas que cubren algunos de los muchos casos de uso de GLN. Son dignos de mención los trabajos de Goldberg et al. 1994 [26], el primer sistema GLN comercial producido en los años noventa, donde se aplica GLN para producir pronósticos del tiempo a partir de un sistema previo conocido como FPA, y el trabajo de Cawsey et al. en 1995 [27] donde se procede a la construcción de un sistema que permitía a los pacientes adquirir información de su historial médico en un formato que un paciente, no familiarizado con la jerga y datos médicos, pudiera entender.

Desde los primeros experimentos en el campo hasta hoy ha habido un gran número de avances en el campo, y, en 2018, A.Gatt y E.Krahmer, llevaron a cabo una extensa recopilación sobre este campo y los métodos que se usan. En concreto, nos es de especial interés la sección tercera de su trabajo en la que tratan la GLN basada en aprendizaje profundo y arquitecturas encoder-decoder [28].

Antes de entrar en detalle en este tipo de arquitecturas, vamos a cubrir brevemente algunos de los algoritmos usados en la generación automática de texto. Uno de los primeros algoritmos surge en 1948 cuando C.E.Shannon propone el uso de cadenas de Markov para la generación de texto [29]. Aplicadas al GLN, el uso de este tipo de modelo permite predecir la siguiente palabra de una frase a partir de la actual a través de las probabilidades de que a una palabra M le siga otra N [30].

Otra técnica empleada para la generación automática de texto se basa en el uso de Redes Neuronales Recurrentes (RNN). Dado que las vamos a explicar en detalle en la siguiente sección, en esta simplemente vamos a mencionar trabajos que hayan dado buenos resultados implementándolas. Mikolov et al. [31] proponen en 2010 un modelo de lenguaje usando RNN para la predicción de palabras en tareas de reconocimiento de voz. En la construcción del modelo, un modelo simple con una sola capa intermedia, el vector de entrada en un tiempo t, $x(t)$, se forma de la concatenación de la representación vectorial de la palabra actual y la salida de de las neuronas de la capa intermedia en el momento $t-1$. Los resultados de este trabajo demuestran que aún entrenando el modelo propuesto sobre menos datos que otros modelos entonces estado del arte, el modelo de Mikolov conseguía mejores resultados a cuanto a reducción de errores a nivel de palabra entre otros.

En estrecha relación con las redes neuronales recurrentes están las redes Long Short-Term Memory (LSTM) cuyo funcionamiento también explicaremos en la siguiente sección. M. Sundermeyer et al. proponen el uso de LSTM para hacer frente al alto nivel de complejidad que supone entrenar una RNN para una tarea de traducción francés-inglés [32]. Los resultados de su experimento demuestran que el uso de una LSTM para esta tarea mejoran sobre el uso de un RNN en cuanto a la asignación de probabilidades mayores a aquellas frases que ha visto ya.

Cubiertos los trabajos de mayor relevancia para nuestro proyecto y viendo que los métodos propuestos en esta sección han dado grandes resultados, en la siguiente sección, explicaremos en detalle el funcionamiento de los métodos que aquí hemos mencionado, así como las arquitecturas encoder-decoder que mencionábamos exponía Gatt [28].

2.3. Deep Learning

Como hemos podido comprobar la sumarización abstractiva es una tarea compleja que ha evolucionado a lo largo del tiempo a medida que nuevas técnicas de deep learning han ido surgiendo. Pero, ¿qué es el deep learning?

El deep learning o aprendizaje profundo es un campo del aprendizaje automático o machine learning, subcampo a su vez de la inteligencia artificial, cuyo interés se basa en la programación de un algoritmo que sea capaz de transformar las entradas en las representaciones que más cercanas sean a la salida esperada [4]. Hablamos de aprendizaje profundo cuando de las representaciones obtenidas se obtienen otras y así de manera sucesiva. La pregunta inmediata que surge es: ¿cómo se obtienen estas representaciones? O lo que es lo mismo, ¿cómo aprende el algoritmo?

El aprendizaje profundo utiliza para aprender redes neuronales. Una Red Neuronal (RN) la componen distintas capas, una capa de input (input layer), una de output (output layer) y las capas intermedias (hidden layers). Hemos creado, con la ayuda de la herramienta creada por A.Lenail [33] la figura 2.3, una representación de una Red Neuronal.

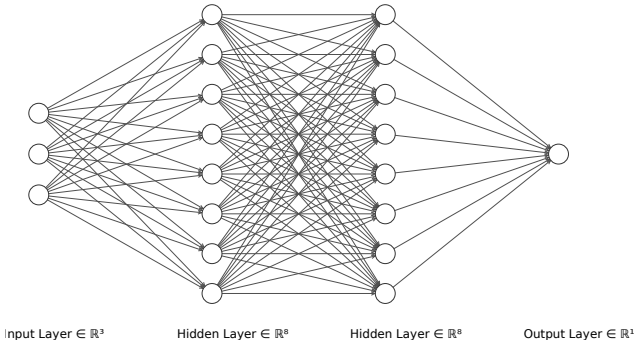


Figura 2.3: Representación Red Neuronal Simple [33]

Las capas que conforman una red, están formadas a su vez por nodos o neuronas. Una neurona de una RN modela de forma simplificada el comportamiento de una neurona cerebral. Estas neuronas, las unidades logísticas de las redes neuronales, reciben un input, una entrada, llevan a cabo las operaciones necesarias y generan un output.

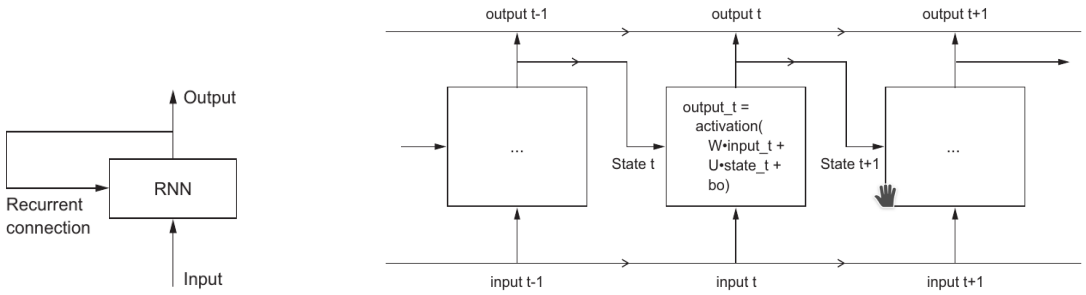
Cuando una Red Neuronal recibe un input, las neuronas llevan a cabo una serie de cálculos de acuerdo con una función de activación y los resultados generados, los pesos calculados de acuerdo con esta función de activación, son pasados a la siguiente capa de la red y así sucesivamente hasta llegar a la última capa de la red. Las redes neuronales que funcionan con este tipo de alimentación entre capas donde el output de la capa i-1 se convierte en el input de la capa i, reciben el nombre de redes neuronales Feedforward.

2.3.1. Redes Neuronales Recurrentes

Para nuestro trabajo nos interesa especialmente un tipo de Red Neuronal cuyo comportamiento difiere ligeramente del explicado en el párrafo anterior. Las Redes Neuronales Recurrentes (RNN) son redes feedforward que guardan un estado, a modo de memoria que combinan con la entrada. Esto es muy útil para tareas como la traducción entre idiomas o la sumarización en los que se trabaja por secuencias, que pueden ser grupos de palabras o caracteres.

En la figura 2.4(a) adjuntamos un esquema de este tipo de redes neuronales en la que se aprecia el bucle que forman las salidas de la red convirtiéndose en una parte de la entrada en el siguiente paso. Se aprecia con mayor detalle en la figura 2.4(b), donde vemos una RNN “desenrollada“, cómo las salidas de las neuronas dentro de la red se convierten de nuevo en parte de la entrada del resto de neuronas.

Este tipo de redes nos es especialmente interesante pues es crucial a la hora de predecir una palabra saber cuál ha sido la anterior, no se trata solo de probabilidades como con los modelos basados en cadenas de Markov, sino también de contexto. Sería interesante saber cómo funciona cada celda de la Red Neuronal Recurrente y qué es esto de la memoria que comentábamos. Decíamos, cuando



(a) Simplificación de una RNN

(b) Desglose de una RNN

Figura 2.4: Esquema del funcionamiento de una Red Neuronal Recurrente de [4]

hablábamos de Red Neuronal que cada neurona recibe una entrada, llevaba acabo un cálculo de acuerdo con una función de activación y pasaban los resultados a la siguiente capa.

En las RNNs este comportamiento se reproduce de igual manera. Digamos que el estado inicial de nuestra Red Neuronal Recurrente es simplemente un vector a 0. Pongamos también, en este caso particular, y para poder basarnos en la imagen, que el momento $t-1$ es el que recibe este estado inicial. Vamos a intentar explicar, con la ayuda de algunos artículos [34] [35] y el capítulo sexto de Goodfellow [36], lo que sucede dentro de la neurona en el momento $t-1$ para el cálculo de la salida.

La neurona $t-1$, recibe la estado de la que ha producido la neurona $t-2$ y procede al cálculo del estado para esa celda:

$$h_{t-1} = f(h_{t-2}, x_{t-1}) \tag{2.2}$$

En esta ecuación f es la función de activación, h_{t-2} es el estado de la celda anterior y x_{t-1} es la entrada o input en el momento $t-1$ que es para el que calculamos el estado.

La ecuación 2.2 también se puede escribir de la manera en la que viene en la imagen 2.4(b). Donde W es el peso de la matriz de estados para la anterior neurona y U es el peso de la capa intermedia para la neurona actual.

$$h_{t-1} = f(W * h_{t-2} + U * x_{t-1}) \tag{2.3}$$

Una vez tenemos el estado de la celda pasamos a calcular la salida de la celda, usando el estado en ese momento y el peso de la capa de salida W.

$$y_{t-1} = W * h_{t-1} \tag{2.4}$$

Para una explicación más detallada y una explicación a fondo de las ecuaciones que rigen estas redes se aconseja leer el trabajo de A.Sherstinsky [37].

Por otro lado, aunque este tipo de redes sean muy útil para el trato de sequencias [38], tienen algunos inconvenientes, tanto por el tipo de memoria que concede, como a la hora de entrenar una Red Neuronal Recurrente. Los problemas que plantean a la hora de entrenarse una RNN han sido estudiados a fondo por autores como Yoshua Bengio, quien expone por primera vez en 1994 [39] los problemas de estas redes con algoritmos de gradiente descendiente y los vuelve a tratar en 2013 [40] en mayor profundidad. En ambos trabajos se trata el problema de desvanecimiento de gradiente y el problema del gradiente explosivo.

El problema del desvanecimiento de gradiente, expone que si se inicializan los pesos de la red con valores demasiado pequeños, debido a los valores que puede tomar la derivada (gradiente) de la función de activación, al aplicar Backpropagation los valores actualizados de la matriz de pesos apenas cambian y esto impide el correcto entrenamiento de la red. El problema del gradiente explosivo provoca también la desestabilización de la red, pero en este caso, debido a que, al actualizar los pesos, estos lleguen a valores tan pequeños que no se puedan seguir actualizando. Por último, el problema que suponen en cuanto a memoria es que estas redes tienen memoria a corto plazo y, si la secuencia entrada es demasiado larga, como es el caso de textos largos, las palabras del principio se irán olvidando.

Long Short-Term Memory

Impulsadas por los inconvenientes que exponíamos de las Redes Neuronales Recurrentes de memoria y de gradiente, surgen alternativas que intentan solventarlos. Son las redes Long Short-Term Memory. Estas redes se basan en el uso de puertas, de ahí que se conozcan como “gated“ RNNs. Como se expone en el Deep Learning Book de Goodfellow [36], las “gated“ RNNs buscan crear caminos en el tiempo cuyas derivadas eviten que el gradiente se desvanezca o explote. Decimos que estas redes se basan en puertas ya que al contrario que con las Redes Neuronales Recurrentes que acumulaban el conocimiento en su memoria sistemáticamente, este tipo de redes permite olvidar información que ya haya sido usada.

Las Long Short-Term Memory, constan de tres tipos de puertas: una puerta de entrada, una puerta de salida y una puerta para olvidar como se puede ver en la figura2.5. Haciendo uso de estas tres puertas, las redes LSTM deciden si una entrada se almacena o no según la importancia que tenga. Por lo general, cuanto más aparezca una entrada en el dataset, la red evitará que estas entradas se olviden. Esto se debe a que el cálculo del estado para una entrada x se lleva a cabo a partir del peso del bucle que depende de la puerta del olvido, dominada por una función sigmoide que le asigna un peso de 1 o 0.

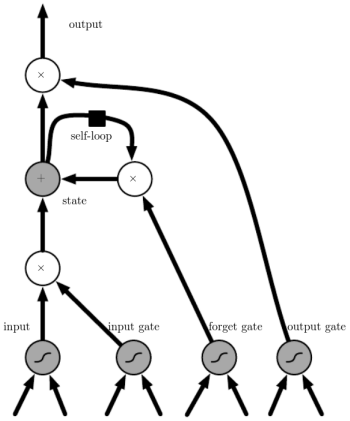


Figura 2.5: Interior de una celda LSTM por Goodfellow en Deep Learning Book [36]

Arquitectura encoder-decoder

Las arquitecturas encoder-decoder se basan en el uso de Redes Neuronales Recurrentes (o sucesores) para crear un modelo comprendido de dos partes: un encoder, que recoge la entrada y la pasa a su forma vectorial, y un decoder, que recibe la salida del encoder y la transforma de vuelta a formato secuencia. Esta arquitectura, también conocida como Seq2Seq, al basarse en Redes Neuronales Recurrentes, permite que se guarden estados y aprendan así el contexto del elemento de la secuencia que están tratando. Uno de los primeros trabajos usando esta arquitectura para una tarea de NLP con éxito fue el de Sutskever et al. [41] usando este modelo para la traducción inglés-francés. En estas arquitecturas el encoder va recibiendo secuencias que va propagando hacia el decoder mientras va actualizando el vector de estados. Este vector de estados que se genera como salida del encoder lo recibe el decoder como entrada para generar las salidas.

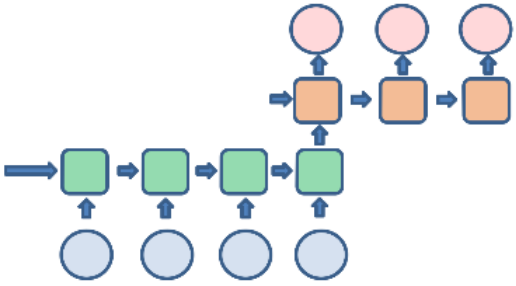


Figura 2.6: Arquitectura encoder-decoder [42]

3

Metodología

Tras la introducción a la sumarización y a algunos de los trabajos del campo, en este capítulo exponemos las decisiones tomadas y métodos implementados para la sumarización extractiva y abstractiva.

3.1. El sumario

En esta primera sección del capítulo abordamos la aplicación práctica de nuestro proyecto a la teoría explicada en la subsección 2.2.1. Más específicamente, hablaremos del dataset del que hemos hecho uso, trataremos los métodos que hemos empleado para generar sumarios extractivos y finalmente, los métodos de evaluación para los resúmenes obtenidos.

3.1.1. Dataset

Como mencionábamos en la introducción, la gran mayoría de los trabajos de sumarización se han realizado en inglés. A la hora de encontrar una compilación de noticias sobre las que trabajar, esto suponía un problema, ya que nuestra intención era llevar la sumarización al castellano.

El dataset por excelencia para la generación de sumarios fue propuesto por Nallapati et al. [38] en 2016. Este dataset, el CNN/Daily Mail, es una recopilación de noticias de la CNN y el Daily Mail y sus resúmenes. Este trabajo tuvo una gran repercusión en el campo y muchísimos proyectos se han basado posteriormente sobre este dataset, tanto de sumarización abstractiva (L.Liu et al. 2018 [43]) como de sumarización extractiva (J.Chen et al. 2016 [44]), en ambos casos produciendo resultados del estado del arte.

No fue tan fácil de encontrar un dataset de noticias en castellano, pero, finalmente, acabamos dando con una compilación de noticias, lo más parecido al ya mencionado dataset CNN/Dailymail de Nallapati et al. [38]. Este dataset, creado por el equipo de Rafael Dueire Lins y Steven J. Simske, recoge noticias del CNN en español para la sumarización extractiva monodocumento y está disponible para trabajos y proyectos de investigación bajo petición a sus creadores. Lins, Simske et al. detallan

los métodos de construcción de este dataset en 'The CNN-Corpus: A Large Textual Corpus for Single-Document Extractive Summarization' [45] y 'The CNN-Corpus in Spanish: a Large Corpus forExtractive Text Summarization in the Spanish Language' [46].

3.1.2. El resumen

Como mencionábamos en el capítulo segundo de esta memoria, la necesidad del sumario surge de la cantidad de información digital existente. Veíamos también, que de entre los distintos tipos de sumarios existentes, para la primera parte de nuestro proyecto, la obtención de resúmenes, nos centrábamos en los sumarios extractivos generales. Esto quiere decir que para la generación de nuestros sumarios, asumimos que los temas de los textos con los que trabajamos (las noticias) son múltiples, e indicamos también, que los sumarios que vamos a producir son el resultado de elegir las frases más importantes de las noticias recibidas como entrada.

En las subsecciones siguientes, explicaremos los dos métodos aplicados para la generación de sumarios extractivos.

Sumarización por peso de frases

El primer algoritmo llevado a cabo para la obtención de sumarios extractivos se basa en la técnica propuesta por Luhn [11] que exponíamos en el segundo capítulo, sección 2.2.1. El algoritmo empleado es bastante sencillo, podemos ver su pseudo-código más abajo en la figura 3.1. Como podemos observar, este algoritmo cuenta con tres métodos principales: *CalculateWordWeights*, *CalculateSentenceWeights* y *GenerateSummary*. Procedemos a explicar la funcionalidad de cada uno de estos métodos a los que llamamos para cada noticia del dataset ya mencionado.

input : noticias del Dataset CNN/DailyMail
output: Lista de resúmenes, cada resúmen una lista con las N frases más relevantes de la noticia

```
1 sumarios ⟵ [ ]
2 foreach noticia ∈ input do
3 |   word_weights ⟵ CalculateWordWeights( noticia )
4 |   sent_weights ⟵ CalculateSentenceWeights( noticia, word_weights )
5 |   sumario ⟵ GenerateSummary( sent_weights, num_frases )
6 |   sumarios ⟵ sumarios + sumario
7 end
```

Algoritmo 3.1

La llamada al método *CalculateWordWeights* sobre cada una de las noticias tiene tres objetivos principales. El primero de ellos es tokenizar la noticia recibida en palabras para poder limpiar la noticia de stop-words y quedarnos simplemente con las palabras clave de ésta. Una vez hecho esto, procedemos a quitar la puntuación y otros símbolos irrelevantes como por ejemplo signos de exclamación o de interrogación. Una vez limpias las palabras, podemos hacer el cálculo de la importancia de cada

palabra. Para ello, primero, contamos la frecuencia con la que cada palabra aparece en la noticia, después, miramos qué palabra aparece más veces, y a continuación, calculamos la frecuencia ponderada respecto a esa palabra.
La salida de este método, *word_weights*, contiene: la palabra, las veces que aparece en la noticia y su frecuencia ponderada. Se puede ver un ejemplo de la salida en el cuadro 3.1.

[['osos', 10, 1.0], ['autoridades', 8, 0.8], ['mujer', 5, 0.5], ['alimentar', 5, 0.5], ['vida', 4, 0.4],...]

Cuadro 3.1: Ejemplo de la salida para una noticia del método*CalculateWordWeights*

Una vez se tienen las frecuencias ponderadas de las palabras en cada noticia, podemos calcular el peso de cada frase usando el método *CalculateSentenceWeight*. Este método simplemente tokeniza la noticia en frases, mira qué palabras contiene que no sean stop-words y va sumando la frecuencia ponderada de las palabras que contiene. Una vez calculada el peso de cada frase, se asocia la posición de la frase en la noticia con su peso, se genera una lista para cada noticia y se devuelve el resultado.

Por último, el método *GenerateSummary* se encarga de recuperar la información generada por el método anterior y seleccionar el número de frases determinado por el argumento *num_frases* con mayor puntuación. Para la implementación de este algoritmo hacemos uso de la librería de Python NLTK [47] para la tokenizar y gestionar los stop-words, y la librería Pickle para guardar los pesos de las palabras y frases.

Sumarización por Pagerank

El segundo algoritmo implementado se basa en el uso de PageRank, el cual también hemos explicado en el segundo capítulo, sección 2.2.1. Además de usar PageRank usamos la similitud coseno, una técnica que quizá nos recuerde ligeramente a los word embeddings.

El pseudo-código de este algoritmo lo encontramos en el cuadrante 3.3. Vamos a desgranarlo poco a poco. Al igual que en el primer algoritmo explicado, en la subsección anterior, necesitamos que las noticias con las que trabajamos estén libres de stop-words, por ello, el primer paso que realizamos es ese a través de la llamada al método *CleanSentences*.

Una vez las noticias carecen de estas palabras pasamos a crear las matrices de similitud para cada noticia. De esta tarea se encarga el método *BuildSimilarityMatrix*, el cual crea para cada noticia, una matriz de tamaño (nfr,nfr), siendo nfr el número de frases que tiene la noticia. Una vez creada la matriz, para cada par de frases de la noticia se calcula la similitud entre estas frases.

La similitud entre frases se calcula creando un vector que represente a cada frase de tamaño (1, $palabras\_total$), siendo $palabras\_total$ el número de palabras que existen en las dos frases eliminando

aquellas repetidas. A continuación, se crea un vector para cada frase y por cada palabra de la frase se suma una unidad en la posición que ocupa esa palabra en $palabras\_totales$. Por último, se devuelve la distancia coseno calculada a través de la librería NLTK [47] y se guarda en la matriz de similitud.
La distancia coseno de las dos frases de la historia representa como de cerca están sus vectores en el espacio vectorial. Esto quiere decir que cuantas más palabras se parezcan y cuanto más se parezcan entre ellas, más cerca se encontrarán los vectores de estas frases.

```
1  Function SentenceSimilarity( sent1, sent2 ) :
2      palabras_totales ⟵ Palabras(sent1) + Palabras(sent2)
3      palabras_totales ⟵ EliminarRepeticiones(palabras_totales)
4      vec1 ⟵ Vector(1, palabras_totales)
5      vec2 ⟵ Vector(1, palabras_totales)
6      for word ∈ sent1 do
7          vec1[palabras_totales.Index(w)] += 1
8      end
9      for word ∈ sent2 do
10         vec2[palabras_totales.Index(w)] += 1
11     end
12     return DistanciaCoseno(vec1, vec2)
13
```

Algoritmo 3.2: Pseudo-código de la función para el cálculo de la similitud entre frases

Una vez ya tenemos las matrices, solo queda aplicar *PageRank* (consultar subsección 2.2.1). Nos valemos para ello de la librería NetworkX [48] que nos permite crear un grafo PageRank a partir de cada matriz. Aplicamos por último los métodos de *Order* y *GenerateSummary*, ordenando el grafo para que las frases con mayor puntuación sean las primeras, y seleccionando cuantas sean necesarias.

```
input : noticias del Dataset CNN/DailyMail
output: Lista de resúmenes, cada resúmen una lista con las N frases más relevantes de la noticia
1  sumarios ⟵ [ ]
2  for noticia ∈ input do
3      clean_sents ⟵ CleanSentences ( noticia )
4      mat_sim ⟵ BuildSimilarityMatrix ( noticia )
5      grafo_sim ⟵ GraphFromMatrix ( mat_sim )
6      pr ⟵ PageRank ( grafo_sim )
7      pr ⟵ Order ( pr )
8      sumario ⟵ GenerateSummary ( pr, num_frases )
9      sumarios ⟵ sumarios + sumario
10 end
```

Algoritmo 3.3: Pseudo-código del algoritmo similitud por coseno

3.1.3. Evaluación del resumen

A la hora de evaluar los resultados obtenidos, necesitábamos una métrica que nos permitiera ver cómo de buenos eran nuestros resúmenes. Por lo general, para la evaluación de sumarios se usa la métrica de ROUGE. Esta métrica trabaja con un sumario de referencia target con el que comparar el obtenido por nuestro algoritmo, pero los sumarios que se nos proporcionaba en nuestro dataset no eran sumarios extractivos, así que tan solo podíamos, a través de esta métrica, evaluar a nivel de palabra la calidad del resumen. Veamos cómo evalúa.

El set de métricas ROUGE, Recall-Oriented Understudy for Gisting Evaluation, se basa en el uso de la precisión y *recall* para evaluar los resultados a través de las unidades que se repiten en el sumario original y el sumario obtenido.

El *Recall* (3.1) mide el número de unidades del sumario original que se encuentran en el sumario obtenido. Las ”unidades” pueden ser las palabras del texto contadas de manera individual, unigramas (ROUGE-1), de dos en dos, bigramas (ROUGE-2) o más (ROUGE-N). Cuanto más cercano a uno sea la puntuación calculada para el n-grama, más se parecerá el sumario obtenido al de referencia.

$$recall = \frac{u\_repetidas}{u\_repetidas + u\_ignoradas} \tag{3.1}$$

La precisión (3.2) evalúa cuantas palabras del sumario obtenido no se encuentran en el original. Indica cómo de relevante es toda la información recogida en el sumario obtenido.

$$precision = \frac{u\_repetidas}{u\_repetidas + u\_equivocadas} \tag{3.2}$$

Para evaluar correctamente los sumarios, ROUGE usa la métrica F-Measure, la media armónica de la precisión y el recall dado por la ecuación 3.3. Cuanto más cercano a uno el resultado mejor será el resultado obtenido.

$$f = 2 * \frac{precision * recall}{precision + recall} \tag{3.3}$$

Además, F-Measure, se usa para una última métrica de ROUGE que nos interesa, la métrica ROUGE-L. Esta métrica se basa en Longest Common Subsequence (LCS) , y busca la longitud de la secuencia máxima coincidente entre la referencia y el sumario obtenido. Esta métrica difiere con respecto a ROUGE-N en el sentido de que aún buscando la subsecuencia más larga común de entre los dos sumarios, ROUGE-L refleja el orden de las unidades de los sumarios lo cual puede alterar radicalmente el significado del sumario obtenido [49].

Como decíamos, ROUGE necesita para evaluar correctamente un resumen, uno de referencia. En nuestro caso como este no existía, a nivel de palabra, no podemos hacer uso de la métrica completa pero sí de ROUGE-1 y ROUGE-2. Con ellas, los resultados obtenidos con estas métricas son para ambos muy pequeños porque la métrica ROUGE se basa para evaluar los sumarios, en el orden de las unidades que lo componen, en la semántica de los textos. Esta métrica de valores pequeños a nuestros resultados porque el orden de las palabras de nuestros sumarios y los de referencia no coinciden. Esto no quiere decir que el significado del sumario difiera del del texto original ni que su calidad sea peor. Por ello proponemos el uso de métricas basadas en la semántica para la evaluación de los sumarios.

Aunque menos comunes, en la última década el interés por evaluar automáticamente los sumarios ha aumentado, contribuyendo a que surjan nuevos métodos para la evaluación de sumarios sin necesidad de referencia. Entre ellos, encontramos estudios que hacían referencia a sistemas complejos como Assessing Summaries without Human reference using ROUGE (ASHuR) [50] y Framework for Evaluating Summaries Automatically (FRESA) [51], que descartamos por la complejidad que les acompañaba.

Habiendo tratado con embeddings, principalmente para la generación de titulares que abordamos en el siguiente capítulo, pensamos que creando el embedding del texto original y el embedding del sumario obtenido podíamos evaluar los resultados obtenidos. Así es como encontramos Flair y Bidirectional Encoder Representations from Transformers (BERT) .

Como se expone en el artículo de A.Akbik et al. ”FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP” [52], Flair es un framework simple diseñado para facilitar el entrenamiento de técnicas NLP estado-del-arte (clasificación de textos, modelos de lenguaje). La idea detrás de Flair es presentar una interfaz simple y unificada para diferentes tipos de embeddings, de palabras y de documentos.

Por otro lado, BERT, presentado por J.Devlin et al. en ”BERT: Pre-training of Deep Bidirectional Transformers forLanguage Understanding” [53], se trata de un modelo que usa un mecanismo de atención para aprender la relación de las palabras del texto en su contexto y que ha dado grandes resultados en distintas tareas de NLP.

De ambas opciones, Flair permitía crear el embedding del texto original entero con distintos embeddings, entre ellos GloVe y BERT. Sin embargo, dada la longitud de nuestros textos originales, de media unas 500 palabras, eran demasiado largos y la librería no los soportaba. Por ese motivo nos decantamos finalmente por BERT.

Para poder usar BERT tuvimos que encontrar librerías que lo implementaran. Al final, usamos la librería Sentence-Transformers [54]. Este librería permite crear embeddings a partir de una lista de frases dada y soportaba distintos idiomas, entre ellos el castellano.

Los pasos seguidos para crear los embeddings son los siguientes:

1.– Creación del modelo sentence-transformers.

2.– Cálculo de los embeddings del texto original.

2.1.– Debido a la entrada esperada por el codificador del modelo, división del texto original en trozos de aproximadamente 200 palabras (cogiendo siempre frases completas).

2.2.– Codificación de los trozos y cálculo del promedio de los embeddings de los trozos.

3.– Cálculo de los embeddings de los sumario obtenidos.

3.1.– Codificación de los embeddings para el resumen generado usando el algoritmo 3.3.

3.2.– Codificación de los embeddings para el resumen generado usando el algoritmo 3.1.

4.– Cálculo de la distancia coseno entre el embedding de cada uno de los resúmenes y el embedding promedio del texto original.

3.2. Generación de titulares

Abordados los procedimientos seguidos para la sumarización extractiva, en esta sección, procedemos a presentar el trabajo llevado a cabo para la generación de sumarios abstractivos cortos a modo de titulares con respecto a lo aprendido en la subsección 2.2.2 así como la sección 2.3.

Más específicamente, hablaremos del dataset del que hemos hecho uso, los embeddings empleados y la creación de las matrices para el entrenamiento del modelo. También cubriremos el modelo construido y los métodos de evaluación de los titulares.

3.2.1. Dataset

Para la tarea de sumarización abstractiva, requeríamos de un dataset, que no solo estuviera en castellano sino que además de las noticias nos proporcionara con los titulares de estas, los targets con los que aprendería nuestro modelo.

El dataset finalmente empleado para esta tarea, proporcionado por la empresa Airenove, consta de 45599 noticias, con sus respectivos titulares, cuerpos de noticia y otra información que no nos es de uso para el proyecto como su fecha de publicación y subtitulares.

A la hora de trabajar con estas noticias, el dataset lo dividimos en tres sets:

- El *test set*, conformado por un $30\,\%$ de las noticias totales, $15{,}500$ aproximadamente. Se usan para evaluar el modelo una vez entrenado.
- El *training set*, formado por el $80\,\%$ de las noticias restantes del *test set*. Son las noticias con las que entrenamos el modelo.
- El *validation set*, el $20\,\%$ de las noticias restantes del *training set*. Estas noticias sirven para ver cómo de bien o mal está aprendiendo el modelo durante el entrenamiento y ajustar en caso de que sea necesario los hiperparámetros.

En resumen, el test set utilizado lo conforman aproximadamente $15{,}500$ noticias, las $30{,}000$ restan-

tes se dividen en 24000 para el set de entrenamiento y $6{,}000$ para el set de validación.

3.2.2. Embeddings

En la subsección 2.2.2 del estado del arte cubríamos los word embeddings y cómo su uso permitía representar una palabra con un vector que recogiera su relación con otras palabras que aparecían en un mismo contexto.

De las dos técnicas para crear word embeddings, GloVe y Word2Vec, durante el proyecto decidimos usar la segunda pues el uso de embeddings GloVe requiere de una gran cantidad de memoria de la que no disponíamos en nuestro entorno de trabajo de Google Colab.

Para crear los embeddings, hacemos uso de la librería Gensim que implementa Word2vec (tanto Skip-gram como CBOW). En nuestro caso, nos decantamos por CBOW, ya que es la arquitectura más rápida para crear los embeddings.
Debemos elegir también el tamaño de los embeddings para cada palabra. Las dimensiones de los embeddings suelen estar comprendidas entre 100 y 500. A lo largo de las numerosas pruebas y entrenamientos llevados a cabo, fuimos experimentando con con estos valores, y concluimos que el tamaño que mejores resultados daba era 200. Así, cada palabra del vocabulario es una matriz fila de de dimensión 200 (o lo que es lo mismo forma $(200,)$).

La creación de los embeddings, necesita de una lista de frases donde sus elementos son cada palabra de las noticias para las que queremos crear los embeddings. Por ello, antes de crear el modelo Word2Vec, creamos las matrices de los textos (pseudo-código en el cuadro 3.4).

```
1  Function CrearMatricesPalabras( articulos, titulos ) :
2      articulos, titulos ⟵ AnyadirTokenStartEnd(articulos, titulos)
3      x ⟵ EsAlfanumerico( Palabras(articulos))
4      y ⟵ EsAlfanumerico( Palabras(titulos))
5      x, y ⟵ Recortar(x, limitex, y, limitey)
6      x, y ⟵ AnyadirPadding(x, y)
7      return x, y
8
```

Algoritmo 3.4: Pseudo-código de la función para el pre-procesamiento de los textos

Este algoritmo recibe los artículos y títulos, y añade un token de inicio de secuencia y fin de secuencia que son necesarios para el modelo Seq2Seq. Después, los textos se parten a nivel de palabra, eliminando todas aquellas que contengan símbolos. Esto nos ayuda a limpiar las palabras de final de línea que contengan guiones entre medias. En este caso de generación de sumarios abstractivos, nos interesa mantener las stop-words ya que sin ellas, nuestros titulares estarían solamente formados por palabras clave. Después, recortamos los textos para que tengan una longitud fija, uno de los requerimientos del modelo encoder-decoder. Los artículos los limitamos a 400 palabras, la longitud media, y

los titulares a 18. Por último, se añade padding a aquellos textos que no lleguen a los límites expuestos. Los artículos llevarán el padding al principio para que la red solo empiece a aprender a partir del token de comienzo ”start”, y los titulares llevarán el padding al final tras la etiqueta de ”end”.

Una vez tenemos las matrices, podemos crear el modelo Word2Vec. En el anexo hemos añadido cómo quedaría una de estas matrices (anexo B.2) basándonos en el comienzo de una noticia (anexo B.1). Con el modelo creado, podemos ver qué vector se corresponde con cada palabra aprendida, cuales son las palabras más cercanas a una dada, y su cercanía a la palabra dada. Se muestra un ejemplo en el cuadro 3.2 con las palabras ’psoe’ (columna izquierda) y ’pp’ (columna derecha).

'pp' $\rightarrow 0{,}8578348159790039$,	*'psoe'* $\rightarrow 0{,}8578348159790039$,
'psc' $\rightarrow 0{,}8377245664596558$,	*'pnv'* $\rightarrow 0{,}7863425016403198$,
'pnv' $\rightarrow 0{,}8339154124259949$,	*'psc'* $\rightarrow 0{,}7822422981262207$,
'nc' $\rightarrow 0{,}7506816387176514$,	*'cs'* $\rightarrow 0{,}7140632271766663$,
'cs' $\rightarrow 0{,}7217929363250732$,	*'senado'* $\rightarrow 0{,}7135229110717773$,
'senado' $\rightarrow 0{,}7076349854469299$,	*'congreso'* $\rightarrow 0{,}7000465393066406$,
'tripartito' $\rightarrow 0{,}7005566954612732$,	*'vox'* $\rightarrow 0{,}6906355619430542$,
'independentismo' $\rightarrow 0{,}6977593898773193$	*'parlament'* $\rightarrow 0{,}6884758472442627$

Cuadro 3.2: Palabras más similares a ’psoe’ y ’pp’ usando Word2Vec

Habiendo creado el modelo Word2Vec, podemos ver que el vocabulario de las noticias de validación y entrenamiento, consta de 150409 palabras distintas. Dado que el vocabulario es extenso, para facilitar el entrenamiento, este se reduce a las palabras más comunes y se eliminan aquellas que tan solo aparezcan una vez, sustituyéndolas en las matrices por la palabra *”unk”*. El vocabulario queda así en 85735 palabras.

Finalmente, a partir de los embeddings, podemos crear las matrices para el entrenamiento del modelo. Contaremos con un total de tres matrices: la matriz de entrada al encoder, *encoder_input_data*, la matriz auxiliar del decoder, *decoder_input_data*, y por último, la matriz con los titulares objetivos *decoder_target_data*.

3.2.3. El modelo

El modelo construido se basa en la arquitectura encoder decoder presentada ya en la subsección 2.3.1. Por eso, en la subsección anterior hacíamos que los textos de entrada tuvieran todos la misma longitud, y que las salidas también lo hicieran. En la figura 3.1 mostramos la esquematización del modelo, pudiendo ver el tamaño de los textos de entrada (400) y salida (18) así como los embeddings (200) representados en este esquema.

Como los embeddings los hemos generado ya y creado las matrices de entrada a la red, nuestro encoder cuenta tan solo con una capa de LSTM de 256 nodos. De no haber sido así, el encoder y

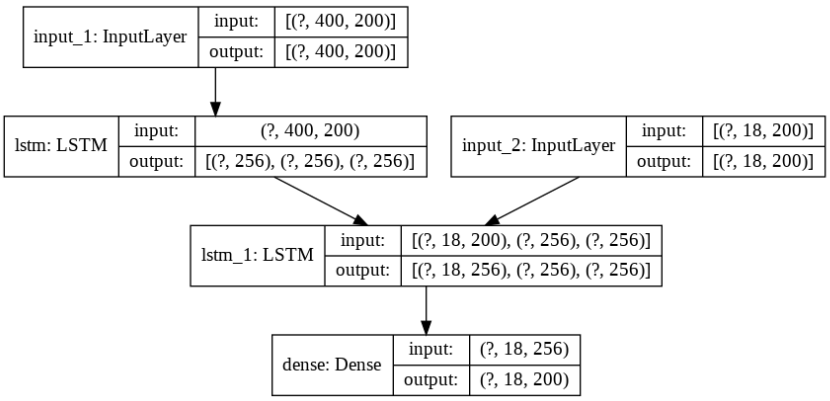


Figura 3.1: Modelo con arquitectura sequence-to-sequence empleado

decoder, contarían con una capa de embedding previa a la LSTM. Esta capa se encarga de aceptar una a una las palabras para cada secuencia de entrada y actualizar el vector de estado. La salida del encoder nos es inútil, pues necesitamos que la secuencia pase por el decoder para producir el sumario, por eso la descartamos y usamos el vector de estados como entrada del decoder.

En cuanto al decoder, este es ligeramente más complejo. Cuenta, en primera instancia, con una capa LSTM cuyo estado inicial es el estado final del encoder. Esta capa, a parte del estado inicial, recibe como entrada la matriz *decoder_input_data*, ayudando a la red en el aprendizaje haciendo uso de la una técnica conocida como teacher forcing. Esta técnica lo que hace es sustituir la salida y_t de una neurona por la salida esperada en ese momento t como entrada en el momento $t+1$ para que los estados calculados a partir de ese momento no se basen en una salida potencialmente errónea.
La segunda capa del decoder es una capa densa, una capa totalmente conectada donde todos sus nodos están conectados a todos los nodos de la capa anterior. Esta segunda capa genera una salida usando la ecuación 3.4 sobre la entrada que recibe, con los pesos correspondientes y el bias aprendido y una función de activación. Esta función de activación es, en nuestro modelo, la función linear, que usamos para redimensionar ya que la capa anterior, la de LSTM consta de 256 nodos y cada palabra es un vector de tamaño $(200,)$.

$$salida = activacion(dot(entrada, pesos) + bias) \tag{3.4}$$

Una vez definidos el encoder y decoder, tan solo creamos el modelo, indicando cuales van a ser sus entradas y cual su salida. La entrada del modelo es doble: la del encoder con los textos de las noticias, y la del decoder, para hacer teacher forcing. La salida será única sin embargo, la salida del decoder serán las predicciones del modelo.

En los primeros entrenamientos la red estaba dando muchos problemas de overfitting, resultados muy buenos para el set de entrenamiento pero valores de pérdida para el set de validación exce-

sivamente altos, señales de que la estaba sobreaprendiento. Para combatir el overfitting, añadimos un regularizador L2 a las capas LSTMs y densas del modelo. Este regularizador simplifica el modelo penalizando la matriz de pesos en el factor indicado.

3.2.4. Evaluación

Dado que los titulares no son más que sumarios abstractivos cortos, y que nuestro dataset cuenta con titulares de referencia, para la evaluación de los titulares haremos uso de la ya explicada (sección 3.1.3) métrica ROUGE.

4

Resultados

Siguiendo la misma estructura del capítulo tercero, procedemos a exponer los resultados obtenidos.

Analizaremos en primer lugar los sumarios extractivos obtenidos siguiendo la metodología explicada en la sección 3.1.2, y finalizaremos con los titulares resultantes de la aplicación del entrenamiento del modelo expuesto en la sección 3.2.3.

4.1. Análisis del dataset para sumarios extractivos

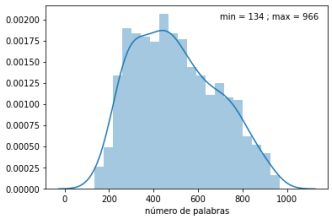
Presentamos a continuación la distribución de las noticias por las categorías en las que se dividen: nacional, deportes, mundo, tecnología, salud, planetaCNN y entretenimiento.

Categorías	Nacional	Deportes	Mundo	Tecnología	Salud	PlanetaCNN	Entretenimiento	Total
Nº de noticias	167	122	116	108	94	70	57	734

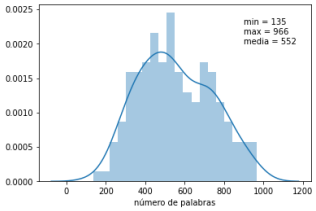
Tabla 4.1: Análisis del dataset CNN usado para sumarización extractiva

Haciendo uso del histograma 4.1(a) podemos ver la mayoría de las noticias tiene una longitud de entre 300 y 600 palabras. El número medio de palabras para este dataset es de 500.26 palabras, siendo el máximo encontrado 966 y el mínimo 134.

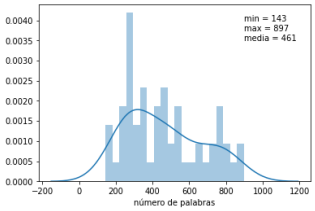
En la figura 4.1 se puede ver ese mismo histograma filtrando por la categoría de la noticia. Podemos ver que, exceptuando los histogramas correspondientes con las categorías de ”entretenimiento” y ”planetaCNN”, para la mayoría de las categorías la gran mayoría de las historias tienen una misma longitud (en torno a 500 para ”nacional” o 600 para ”salud”), mientras que para ”entretenmiento” y ”planetaCNN” las historias tienen longitudes más dispares (dos picos en vez de uno en las gráficas y curva más suave).



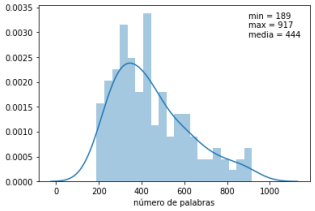
(a) Todas las categorías



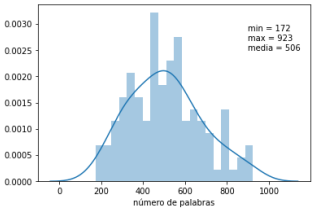
(b) Categoría ”nacional”



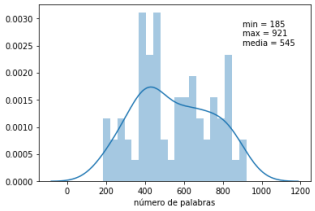
(c) Categoría ”entretenimiento”



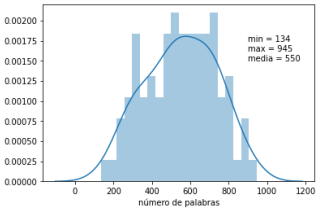
(d) Categoría ”deportes”



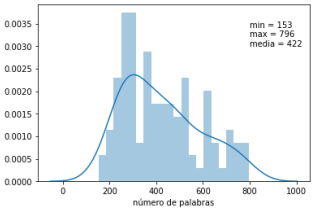
(e) Categoría ”mundo”



(f) Categoría ”planetaCNN”



(g) Categoría ”salud”



(h) Categoría ”Tecnología”

Figura 4.1: Histogramas que muestran las longitudes de las noticias del dataset CNN según sus categorías.

4.2. Sumario extractivo

4.2.1. Ejemplos de sumarios obtenidos

Vamos a ver algunos de los sumarios obtenidos usando los algoritmos 3.1 y 3.3.

Así es como Google procesa los 'emails' que pasan por sus servidores. La empresa fue acusada de violar el derecho a la privacidad de sus usuarios en una corte en Estados Unidos. Nota del editor: Este texto fue actualizado con precisiones sobre los documentos presentados en juicio y su contexto. Google procesa todos los correos electrónicos que pasan por sus servidores de la misma manera en la que lo hace con los correos electrónicos de los usuarios de su servicio de correo, Gmail. Google respondió a una demanda colectiva en la que se le acusa de espiar a los internautas. "Google abre, lee y adquiere ilegalmente contenido privado de los correos electrónicos de la gente", según la acusación de varios denunciantes. En su respuesta, Google apuntó, con respecto a los no-usuarios de Gmail: "Al igual que el remitente que envía una carta a un colega de negocios no puede sorprenderse de que la asistente del destinatario abra la carta, las personas que usan email basado en la web hoy en día no pueden sorprenderse de que sus emails sean procesados por el proveedor de email del destinatario durante el transcurso de la entrega. De hecho, 'una persona no puede tener expectativas legítimas de privacidad en información que voluntariamente entrega a terceras personas'". Con ello, Google se refería a que cualquier email que pasa por sus servidores (por ejemplo, un correo que un usuario de Yahoo! Mail envía a un usuario de Gmail) será procesado de forma similar a como la empresa procesa la correspondencia de los usuarios de Gmail, mediante técnicas como la filtración de mensajes basura (spam) y la configuración de anuncios, indica el sitio especializado en tecnología Mashable. Google envió este comunicado a Mashable: "Tomamos muy seriamente la privacidad y la seguridad de nuestros usuarios. Los reportes recientes que afirman lo contrario son falsos. Hemos construido para Gmail aplicaciones de seguridad y privacidad líderes en la industria, y no importan quién envíe un email a un usuario de Gmail, esas protecciones aplican". Los abogados de Google presentaron el documento, de 39 páginas, este martes ante los tribunales de San José, California. La empresa de internet manifiesta que el escaneado automático (no humano) de correos electrónicos es el procedimiento ordinario en el intercambio de mensajes a través de Gmail, una información que sirve para optimizar la publicidad que recibe el usuario, como lo especifican los términos y condiciones de su servicio de correos. El director del proyectode protección a la privacidad Privacy Project, de la organización de protección al consumidor, Consumer Watchdog, dijo que Google "admitió finalmente que no respeta la privacidad", e invitó a quienes estén interesados en mantener sus comunicaciones en privado a que no usen Gmail. Con información de EFE.

Cuadro 4.1: Texto original de una noticia del dataset

La empresa fue acusada de violar el derecho a la privacidad de sus usuarios en una corte en Estados Unidos. Nota del editor: Este texto fue actualizado con precisiones sobre los documentos presentados en juicio y su contexto . Google respondió a una demanda colectiva en la que se le acusa de espiar a los internautas. "Google abre, lee y adquiere ilegalmente contenido privado de los correos electrónicos de la gente", según la acusación de varios denunciantes.

Cuadro 4.2: Resultados obtenidos por el algoritmo 3.1 para la noticia con texto original mostrado en 4.1

Google respondió a una demanda colectiva en la que se le acusa de espiar a los internautas. En su respuesta, Google apuntó, con respecto a los no-usuarios de Gmail: .Al igual que el remitente que envía una carta a un colega de negocios no puede sorprenderse de que la asistente del destinatario abra la carta, las personas que usan email basado en la web hoy en día no pueden sorprenderse de que sus emails sean procesados por el proveedor de email del destinatario durante el transcurso de la entrega. Mail envía a un usuario de Gmail) será procesado de forma similar a como la empresa procesa la correspondencia de los usuarios de Gmail, mediante técnicas como la filtración de mensajes basura (spam) y la configuración de anuncios, indica el sitio especializado en tecnología Mashable. La empresa de internet manifiesta que el escaneado automático (no humano) de correos electrónicos es el procedimiento ordinario en el intercambio de mensajes a través de Gmail, una información que sirve para optimizar la publicidad que recibe el usuario, como lo especifican los términos y condiciones de su servicio de correos.

Cuadro 4.3: Resultados obtenidos por el algoritmo 3.3 para la noticia con texto original mostrado en 4.1

El 71 % de crímenes contra periodistas en México están impunes: CNDH. El órgano de derechos humanos consideró "de importancia capital.ªbatir la impunidad tras la implementación del mecanismo de protección. De los crímenes contra periodistas y medios de comunicación que la Comisión Nacional de Derechos Humanos (CNDH) ha registrado, el 71 % de ellos siguen impunes, informó este jueves el órgano. Desde 2000, la Comisión ha registrado82 casos de homicidio de periodistas, 16 desapariciones y 28 atentadoscontra medios de comunicación, de los cuales 19 % han sido investigados y solo 7 % derivaron en una condena. "La CNDH considera de importancia capital abatir la impunidad, sobre todo después de la reforma que permitió la creación de unMecanismo para la Protección de Personas Defensoras de Derechos Humanos y Periodistas", informó la CNDH en un comunicado. El mecanismo referido por el órgano fue establecido a través la Ley para la Protección de Personas de Derechos Humanos y Periodistas, que entró en vigor el 25 de junio pasado con su publicación en la Gaceta Oficial de la Federación. El mecanismo busca "que el Estado atienda su responsabilidad fundamental de proteger, promover y garantizar los derechos humanos", según establece la ley. "La impunidad obedece, en gran medida, al hecho de que las autoridades encargadas de integrar las averiguaciones previas, incumplen su obligación de investigar y recabar pruebas para llegar a la verdad de los hechos", señaló la CNDH. El órgano que dirige Raúl Plascencia Villanueva consideró "fundamental atacar el problema de fondo"para evitar más ataques contra periodistas. El señalamiento de la CNDH ocurre después de llamados similares por parte de activistas y órganos internacionales. El jueves pasado, la organización Article 19 y la Alta Comisionada de Naciones Unidas para los Derechos Humanos coincidieron en queel gobierno mexicano ha fallado en dar protección a los periodistas. "Los protocolos y metodologías de las procuradurías para abordar este tipo de delitos no encajan con la urgencia y necesidad y el clamor del gremio periodístico de siempre mantener como una poderosa hipótesis de investigación, tal vez la primera, el que estas agresiones, desapariciones y asesinatos se están dando en razón de la actividad periodística", dijo a CNNMéxico Javier Hernández, representante de la ONU. Tras la entrada en vigor de la ley y del mecanismo de protección, la CNDH emitió una lista de recomendaciones para las autoridadesconsideraran en sus acciones para dar seguridad a periodistas. La organizaciónAmnistía Internacional (AI) también ha denunciado las omisiones y fallasdel gobierno mexicano en los esfuerzos por proteger a periodistas. En mayo pasado, en su informe anual sobre la situación de los derechos humanos en el mundo, señaló que "la provisión de protección a defensores y defensoras era a menudo lenta, burocrática e inadecuada". A mediados de mayo,expertos de Naciones Unidas y de la Comisión Interamericana de Derechos Humanos (CIDH) emitieron un comunicadopidiendo al gobierno mexicano un alto a la muerte de periodistas en el país. Estos llamados al gobierno mexicano se registraron luego de que en la semana del 28 de abril al 3 de mayo, cuatro periodistas fueron asesinados:Regina Martínez, del semanarioProceso, y los fotógrafos Gabriel Huge, Guillermo Luna y Esteban Rodríguez. Luna Varela era fotógrafo en veracruznews. mx, Esteban Rodríguez deNotiver, y Huge "se dedicaba a actividades particulares", detalló la Procuraduría General de Justicia de Veracruz. "Los asesinatos y amenazas repetidamente sufridos por defensores de derechos y periodistas en México deben detenerse inmediatamente", informaron los expertos de la CIDH. El 14 de juniofue asesinado, también en Veracruz, Víctor Báez, periodista del diarioMilenioen Xalapa. El último crimen de este tipo ocurrió el martes pasado, cuandofue reportado como desaparecido Miguel Morales Estrada, fotógrafo delDiario de Poza Rica, también en Veracruz. El gobierno deVeracruz se comprometió en mayo pasado a esclarecer los crímenescontra periodistas en el estado .

Cuadro 4.4: Texto original de una noticia del dataset

Desde 2000, la Comisión ha registrado82 casos de homicidio de periodistas, 16 desapariciones y 28 atentadoscontra medios de comunicación, de los cuales 19 % han sido investigados y solo 7 % derivaron en una condena. "La CNDH considera de importancia capital abatir la impunidad, sobre todo después de la reforma que permitió la creación de unMecanismo para la Protección de Personas Defensoras de Derechos Humanos y Periodistas", informó la CNDH en un comunicado. El mecanismo referido por el órgano fue establecido a través la Ley para la Protección de Personas de Derechos Humanos y Periodistas, que entró en vigor el 25 de junio pasado con su publicación en la Gaceta Oficial de la Federación. El señalamiento de la CNDH ocurre después de llamados similares por parte de activistas y órganos internacionales.

Cuadro 4.5: Resultados obtenidos por el algoritmo 3.1 para la noticia con texto original mostrado en 4.4

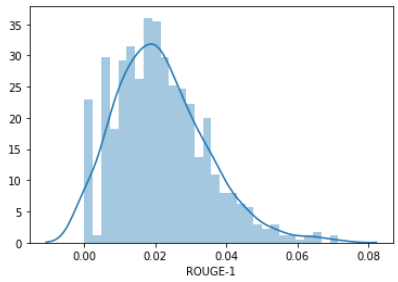
"La CNDH considera de importancia capital abatir la impunidad, sobre todo después de la reforma que permitió la creación de unMecanismo para la Protección de Personas Defensoras de Derechos Humanos y Periodistas", informó la CNDH en un comunicado. El jueves pasado, la organización Article 19 y la Alta Comisionada de Naciones Unidas para los Derechos Humanos coincidieron en queel gobierno mexicano ha fallado en dar protección a los periodistas. "Los protocolos y metodologías de las procuradurías para abordar este tipo de delitos no encajan con la urgencia y necesidad y el clamor del gremio periodístico de siempre mantener como una poderosa hipótesis de investigación, tal vez la primera, el que estas agresiones, desapariciones y asesinatos se están dando en razón de la actividad periodística", dijo a CNNMéxico Javier Hernández, representante de la ONU. Tras la entrada en vigor de la ley y del mecanismo de protección, la CNDH emitió una lista de recomendaciones para las autoridadesconsideraran en sus acciones para dar seguridad a periodistas.

Cuadro 4.6: Resultados obtenidos por el algoritmo 3.3 para la noticia con texto original mostrado en 4.4

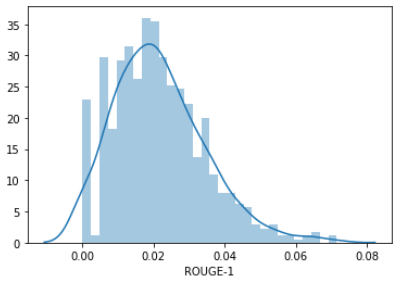
4.2.2. Evaluación con la distancia de ROUGE

En esta sección estudiaremos los valores de la métrica de ROUGE para los sumarios extractivos obtenidos. Recordamos que debido a las peculiaridades del dataset solo podemos estudiar ROUGE-1 y ROUGE-2 pues los sumarios de referencia son extractivos a nivel de palabra.

En los histogramas 4.2(a) y 4.2(b) podemos ver la evaluación de acuerdo con ROUGE-1 de los algoritmos 3.1 y 3.3 respectivamente. Observamos que los valores que estos histogramas devuelven son muy cercanos a cero, implicando que es poca la superposición de palabras (monograma) entre el sumario de referencia y el obtenido. La evaluación de superposición de bigramas (ROUGE-2) de estos algoritmos demuestra valores aún más pequeños. ¿A qué se debe esto?



(a) Algoritmo 3.1



(b) Algoritmo 3.3

Figura 4.2: Evaluación ROUGE-1 sobre los resultados de los algoritmos 3.1 y 3.3

La métrica ROUGE se basa para evaluar los sumarios, en el orden de las palabras y conjunto de palabras. Es decir, ROUGE, se basa en la semántica de los textos para evaluarlos.
Que esta métrica de valores pequeños a nuestros resultados solamente quiere decir, que el orden de las palabras de nuestros sumarios y los de referencia no coinciden. Sin embargo, que este orden no coincida, no quiere decir que el significado del sumario difiera del del texto original ni que su calidad sea peor. Por ello, debemos buscar una métrica basada en la semántica de los textos y que permita evaluar si el significado del sumario se mantiene aunque el texto sea más corto que la noticia original.

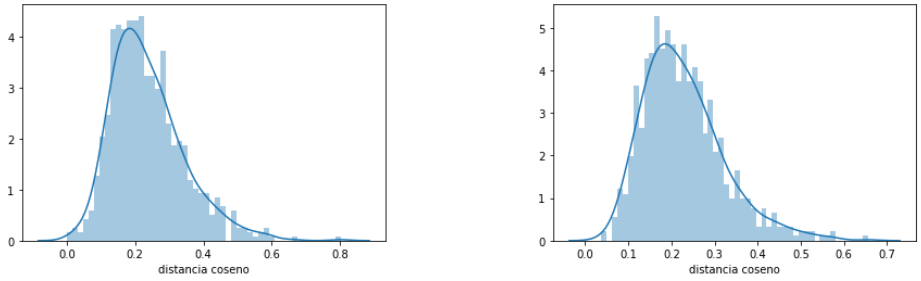
4.2.3. Evaluación con la distancia basada en Bert

BERT es precisamente una métrica que permite evaluar la diferencia semántica entre dos textos dados, uno de referencia, y el obtenido a través de sus embeddings. En esta sección vamos a analizar los resultados obtenidos usando esta métrica.

En primera instancia, vamos a estudiar los resultados obtenidos por el algoritmo 3.1. En la gráfica de la figura 4.3(a), se muestran las distancias coseno del embedding de cada sumario obtenido con respecto al embedding del textos originales. Se observa en la gráfica que la distancia media ronda el valor $0{,}20$, siendo la media de estas distancias $0{,}24$. Esto quiere decir que por lo general, los vectores

que representan los textos originales y los vectores que representan los resultados obtenidos forman un ángulo bastante pequeño entre ellos y los resultados obtenidos son adecuados.

Se muestra una gráfica similar (4.3(b)) para los resultados del segundo algoritmo implementado, el algoritmo 3.3. En este caso la media de las distancias entre los textos originales y los sumarios es ligeramente mayor con un valor de $0{,}23$. Parece a simple vista que este método, sin embargo, produce resultados ligeramente más dispersos.



(a) Algoritmo 3.1 (b) Algoritmo 3.3

Figura 4.3: Gráfica de las distancias coseno entre los sumarios obtenidos con los algoritmos 3.1 y 3.3 y los textos originales usando BERT para la evaluación

Dentro de las distancias si filtramos según la categoría a la que pertenece la noticia y buscamos los mejores y peores resultados, obtenemos la tabla 4.2. No podemos incluir las veintiocho noticias que se corresponden con esas evaluaciones BERT porque no cabría en esta memoria, pero sí las conclusiones que hemos sacado de estudiarlos.

El propósito de la tabla era ver si aquello que causaba peores o mejores resultados para una categoría no afectaban a otra distinta. Por lo que hemos visto, no es así, sino que los mismos problemas afectan a todas las categorías. Para el primer algoritmo, los sumarios peor evaluados surgen de textos que tienen una gran cantidad de números en ellos. Ambos algoritmos dan peores resultados cuando la noticia contiene muchas citas.
También podemos ver gracias a la tabla que las medias de evaluación con BERT por categorías tienen una desviación típica pequeña con lo que todos los resúmenes tienen una puntuación por BERT cercana a esta.

Por último, en la figura 4.4 presentamos la disposición de estas distancias, siendo el eje horizontal el correspondiente con las distancias resultantes del primer algoritmo y el eje vertical las del segundo. La diagonal nos permite de manera visual ver qué método da distancias menores. Numéricamente, los algoritmos son similares, el 50 % de noticias aparecen bajo la diagonal (da mejores resultados el algoritmo 3.1).

En el anexo A se han añadido algunos ejemplos de los sumarios obtenidos y los textos originales de los que provienen.

Categoría	Mejor resultado	Peor resultado	Media	Algoritmo
Nacional	0,037	0,545	$0{,}253 \pm 0{,}099$	3.1
Nacional	0,080	0,571	$0{,}246 \pm 0{,}092$	3.3
Deportes	0,016	0,803	$0{,}224 \pm 0{,}116$	3.1
Deportes	0,065	0,514	$0{,}205 \pm 0{,}078$	3.3
Mundo	0,027	0,608	$0{,}219 \pm 0{,}099$	3.1
Mundo	0,037	0,656	$0{,}222 \pm 0{,}100$	3.3
Tecnología	0,055	0,511	$0{,}218 \pm 0{,}089$	3.1
Tecnología	0,085	0,512	$0{,}206 \pm 0{,}086$	3.3
Salud	0,037	0,660	$0{,}262 \pm 0{,}131$	3.1
Salud	0,097	0,578	$0{,}253 \pm 0{,}099$	3.3
PlanetaCNN	0,029	0,536	$0{,}251 \pm 0{,}095$	3.1
PlanetaCNN	0,075	0,504	$0{,}239 \pm 0{,}092$	3.3
Entretenimiento	0,029	0,522	$0{,}233 \pm 0{,}110$	3.1
Entretenimiento	0,040	0,565	$0{,}226 \pm 0{,}094$	3.3

Tabla 4.2: Mejor y peor resultados filtrando por categoría para los dos algoritmos implementados usando BERT

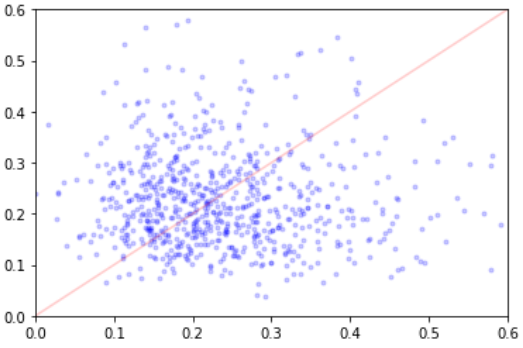


Figura 4.4: Comparación de los resultados obtenidos con los algoritmos 3.1 y 3.3 usando BERT para crear los embeddings de los textos originales y los resultados obtenidos

Concretamente, el primer ejemplo A.1, muestra un sumario que ha obtenido de acuerdo con la métrica de BERT un resultado entorno a la media, para el algoritmo primero la distancia coseno obtenida ha sido de $0{,}241$ y para el segundo la distancia es de $0{,}195$. En los resultados podemos ver que dos de las frases obtenidas se repiten, esto es algo que sucede en algunos de los resúmenes cuando el texto de entrada no es excesivamente largo.
El segundo ejemplo A.2 del anexo, es precisamente el texto que para el algoritmo 3.1.2 da peores resultados dentro de la categoría de deportes. El texto original contiene bastantes números (resultados deportivos) y esto provoca que el algoritmo 3.1 conceda un mayor peso a estas cifras frente a otras palabras lo que genera un resultado nefasto (distancia de $0{,}80$). En comparación, el segundo algoritmo, devuelve un sumario infinitamente superior, con una distancia de $0{,}27$ (aunque aún por encima de la media).

Otras observaciones que hemos podido hacer analizando los resultados son que al tratarse de sumarios extractivos, con frecuencia se eligen frases que hacen referencia a otras que les preceden en el texto original y se pierde parte del significado si no se puede extraer del contexto.
También que cuando las noticias contienen entrevistas, es complicado entender los sumarios porque, por lo general, se queda con las respuestas y no con las preguntas. Además como es lógico al ser extractivo, no reformula la frase y uno debe basarse en sus conocimientos previos del titular.
Por último, el algoritmo 3.1 tiende a recalcar mucho un tipo de información en concreto y si hay una temática secundaria la ignora, mientras que el algoritmo basado en similitud coseno, suele generalizar más el tipo de información que recupera y esto le da un contenido más similar al que daría un agente humano aunque la generación de matrices resulta ser un proceso costoso a nivel de memoria.

4.3. Análisis del dataset para sumarios abstractivos

En el dataset empleado para la generación de titulares, las noticias vienen separadas por el periódico de origen, siendo estas: ”El Mundo”, ”El Pais”, ”ABC”, ”La Vanguardia”, ”Canarias7”, ”El Día”, ”Diario de Avisos”, ”La provincia”, ”As Canarias”, ”El Marca”. El período que más noticias aporta es ”La provincia”.

En la tabla siguiente, mostramos el número de noticias que aporta cada periódico así como la longitud media de esas noticias. La longitud media de las noticias total es de $419{,}54$ palabras.

Periódico	Mundo	País	ABC	Vanguardia	Canarias7	El día	DdA	Provincia	As Canarias	Marca
Nº de noticias	1920	3349	2710	2087	7566	7292	6858	8306	3553	1958
Longitud media	592,03	571,84	500,08	496,32	371,08	442,16	300,29	481,62	261,63	340,39

Tabla 4.3: Análisis del dataset usado para sumarización abstractiva

4.4. Titulares

Una vez el modelo construido, llevamos a cabo un gran número de pruebas para ir analizando los resultados y determinar y ajustar los hiper-parámetros del modelo.

Los primeros entrenamientos nos ayudaron a ver que nuestro modelo necesitaba de regularización para evitar el overfitting. También que la pérdida del modelo era menor con embeddings de 200. Hicimos también pruebas con el número de nodos de las capas LSTM, incrementando su número pero esto solo dio lugar a que la red tardara excesivamente en entrenar sin mejorar los resultados.

Una vez fijados los hiper-parámetros de la red pudimos entrenar el modelo. La figura 4.5 muestra cómo a lo largo del entrenamiento el valor de la pérdida sobre el set de entrenamiento del modelo va disminuyendo y que la pérdida sobre el set de validación también lo hace. Las gráficas previas a la introducción del parámetro de regularización mostraban una línea para la pérdida sobre el set de entrenamiento similar a la que vemos en esta imagen, sin embargo, la línea de la pérdida sobre el set de validación incrementaba al poco de comenzar el entrenamiento.

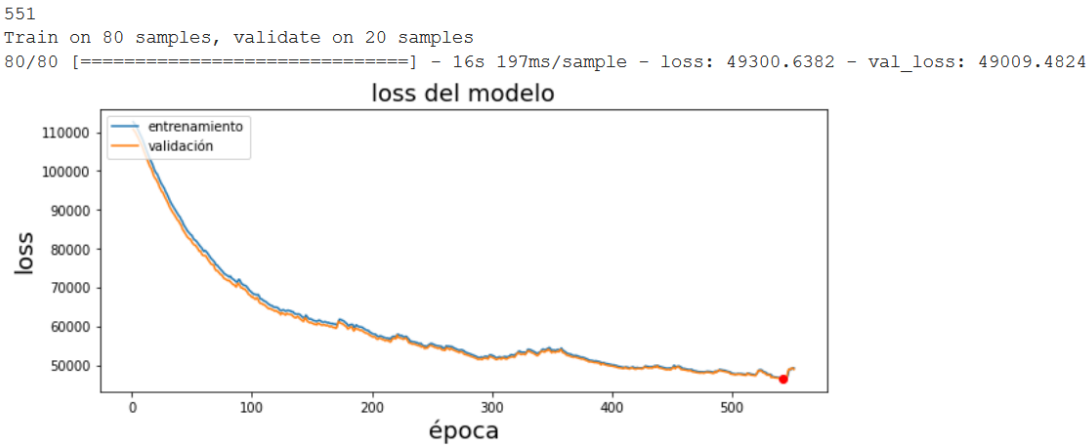


Figura 4.5: Gráfica del entrenamiento del modelo en la época 500

La gráfica anterior se corresponde con los resultados obtenidos hasta la época 551 del entrenamiento del modelo, que al tratarse de una prueba se lleva a cabo sobre 100 noticias del conjunto de entrenamiento. La imagen nos proporciona además los siguientes datos:

- El modelo está entrenando sobre 80 noticias y validando sobre las 20 restantes.
  Esto sigue los porcentajes expuestos en la subsección 3.2.1.
- El entrenamiento para la época mostrada ha durado 16 segundos.
- Para la época 551, la pérdida calculada sobre el conjunto de entrenamiento es de 49300.
- Para la época 551, la pérdida calculada sobre el conjunto de validación es de 49009.

Durante los entrenamientos y pruebas del modelo, lo que ha sido más limitante, sin duda, es el tiempo de entrenamiento necesario para entrenar la red. La gráfica anterior representa tan solo el tiempo de entrenamiento necesario para entrenar sobre un dataset reducido de 100 noticias. El tiempo

necesario para entrenar la red sobre el conjunto entero de entrenamiento sería demasiado como para abordarlo en Google Colab donde contamos con una GPU sin la cual el proceso hubiera sido aún más lento.

De hecho, si probamos a entrenar sobre un conjunto de 1000 noticias, diez veces mayor que el anterior, el tiempo por época pasa de 16 segundos a 19 segundos. Teniendo en cuanta que el vocabulario es obviamente mayor, los valores de pérdida aumentan y la red tarda más en llegar a los valores presentados en la gráfica. Esto tan solo con 1000 noticias, recordamos que el conjunto de entrenamiento consta de 30.000 noticias y el modelo tarda demasiado tiempo en converger como para poder mostrar sus resultados definitivos. Algunas de las predicciones hechas sobre el conjunto de test se pueden ver en los cuadros 4.7 y 4.8 donde vemos que las predicciones muestran una predominancía de stop-words, ya que por lo general son los conexores en castellano y se usan más, consiguiendo que una palabra producida tenga una mayor probabilidad de ser un stop-word que una palabra clave.

Noticia: *Vox afila los dientes y enseña el colmillo para echarse al cuello del nuevo Gobierno y de sus políticas. Ni le concede el más pequeño margen ni le presupone el más mínimo acierto. Así que beligerancia absoluta y oposición «total y frontal». Santiago Abascal remató la tercera jornada de debate de investidura negando cualquier halo de legitimidad al próximo Ejecutivo de coalición del PSOE y Unidas Podemos, y concluyó que éste no era otra cosa más que un «matrimonio entre la mentira y la traición». Los ataques a los pactos de los socialistas con Pablo Iglesias, ERC, PNV o EH Bildu fueron contundentes. Con el líder de Vox desplegando toda su contundencia y retórica contra la izquierda y el independentismo. Incluida la vinculación directa con los postulados de ETA. Un ejemplo de hasta dónde va a llegar la oposición dura y agresiva que planea hacer Vox y que piensa arremeter contra cualquier gesto con el separatismo catalán o vasco. «La compañía aseguradora de la investidura, del golpe institucional que ustedes están dando, se llama ETA», aseguró Abascal. No fue la única alusión a la banda terrorista. El pacto para la abstención de la izquierda abertzale, necesario para lograr más síes que noes en la investidura, fue presentado también por Abascal como el haber conseguido el «beneplácito de ETA». Luego los 52 diputados de Vox se marcharon cuando intervino Bildu. Pero si algo puso en alerta a Abascal y centralizó su discurso fue que Sánchez está sustentado por los «comunistas» de Unidas Podemos y por el separatismo. Sobre los primeros, de los que avisó llevando la tribuna un libro de memorias de Largo Caballero, dijo que quienes van a «copresidir» España tienen «estrechos vínculos con dictaduras, personajes narcoterroristas y teocracias». Sobre los independentistas –o «golpistas» para Vox– apuntó que Sánchez se había «arrodillado» ante ellos para poder ser presidente cuando había dicho que no lo haría. «Un fraude electoral», viene recalcando. «Esto sólo pueden terminar en el blanqueamiento, la impunidad y el referéndum», advirtió. «Sánchez quiere copresidir un gobierno ilegítimo, porque de la mentira y fraude brutal a los españoles sólo puede nacer la ilegitimidad». Para Abascal son tiempos de «extrema preocupación» y «zozobra».*
Titular real: *«La aseguradora es ETA»*
Titular predicho: *podemos busca unk unk end*

Cuadro 4.7: Predicción sobre el conjunto de test

Noticia: *Lloraba Pablo Iglesias porque el llanto es hoy la única forma de hacer política y porque su pasado de activista acababa de morir para siempre. Lloraba Pablo, y nadie recogía esas lágrimas que podrían curar las desigualdades de este país cuyo nombre no debe nombrarse. Este país en el que a partir de ahora las mujeres no morirán apalizadas y los gays bajarán de las grúas de las que pendían hasta hoy para amarse libremente. Los ricos transferirán sus rentas a los pobres. El llanto de Pablo fecundará los campos yermos por el odio de la derecha y sus primeros efectos fertilizantes se advertían ya en la indumentaria de Monedero, al que le ha florecido un traje con corbata. Los cielos han sido asaltados y conviene moverse por ellos como si fuéramos ángeles habituales. No es que Vallecas quede lejos: es que en Galapagar no caben toda una vicepresidencia y cuatro ministerios. Pero conservemos la llaneza publicitaria en la histórica hora, camaradas. Mantengamos el tuteo: «Pedro, te deseo que tengas el mejor tono pero también la mayor firmeza democrática». O sea, golpeemos todo lo duro que nos deje la aritmética, pero que siga pareciendo que somos las víctimas. Leninismo de manual de resistencia. El Pedro al que se refería el vicepresidente investido es un señor que accedió al Hemiciclo al tiempo que su madre y su esposa tomaban asiento en la tribuna de invitados. Era el día grande del chico. ¡Cómo se aplaudía a sí mismo al terminar el recuento, en entrañable posesión del juguete que cinco veces le negaron! No despertéis su narcisismo satisfecho con agoreras profecías sobre la estabilidad, que le importa el mismo comino que a ERC. Al fin es un presidente como los demás. Lloraba Bassa por su hermana presa, lloraba Lastra por su jefe guapo y lloraba España por las plegarias atendidas en el jardín de infancia de la ambición. Entre los testigos no faltó Tezanos, orgulloso de su farsa autocumplida, Iceta sonreía al lado de la hosca Susana y Pepu miraba a todas partes, supongo que buscando el aro. Lejos de la llorera, en el ámbito adulto, algunas frases pronunciadas como si la razón no fuera una antigualla y la Constitución despertara todavía alguna simpatía. «Ser ultra es rebasar los límites y sus socios los han rebasado todos. La ley no es una deriva sino el precio de la libertad. Su única patria es usted, señor Sánchez. Es hombre de paja del nacionalismo», trató de explicarle Casado. «Había una alternativa pero usted desprecia a los constitucionalistas. Cava trincheras porque necesita la gasolina del extremismo», le desnudó Arrimadas, que puso al candidato ante el espejo de quien era en campaña, como si ante los espejos nuestro niño no viera siempre al mismo galán de telenovela, igual de irresistible con cada look. En cuanto al tercer partido por esca- ños, aún no sabemos si Abascal es un problema para Vox o Vox lo es para Abascal. Empezó condenando un crimen machista y luego acertó en la descripción de una estrategia ilegítima que acaparó el poder para ganar las elecciones y no al revés. Pero los gritos extemporáneos y las pataletas grotescas de su bancada no ayudan nada a consolidar el liderazgo institucional que Abascal necesita. A lo que ayudan, y mucho, es a consolidar el mito antifascista que será el único programa de la precaria CoPro (Coalición Progresista), el único pegamento que puede hacerla durar. Ana Oramas habló para otra época, una en que el raciocinio aún podía desafiar la disonancia cognitiva del diputado o el votante. Ni era una villana antes ni soy una heroína ahora, vino a decir: voto en conciencia y pido perdón a mi partido porque debí convencerle de la idoneidad de mi posición. Nadie en esa Cámara completamente polarizada supo aplaudirla, naturalmente. Sí fue muy celebrado –por todos aquellos que no se resignan a olvidar a los asesinados por ETA– el diputado de UPN que limpió el pus revolucionario del grano juvenil que se explotó en directo el orador de Bildu, menciones al Che y la Pasionaria incluidas. Por su parte, el PSOE aplaudió mucho al señor de Teruel, que tiene toda la pinta de creerse que los aplausos llenan la España vacía. También fue muy aplaudido el Gran Recogedor, don Aitor Esteban, que casi estropea la primera comunión de Pedro con este aviso: «Mañana empieza lo difícil». Y hubo minutejos de gloria residual para portavoces variopintos como el del BNG, entrañablemente concentrado en decir «Galiza» –consciente de que en el momento en que se le escapara «Galicia» se jodería el sueño de Castelao–, o como Errejón, que volvió a incurrir en el enésimo error de cálculo cuando en su fantasía comunitarista suplicó no caer en la imagen de un Gobierno contra la mitad de los españoles, que es exactamente lo que es...*[truncada]*

Titular real: *El viaje de Iglesias de la cal al confeti*

Titular predicho: *el el unk unk los end*

Cuadro 4.8: Predicción sobre el conjunto de test

Resulta interesante ver los resultados obtenidos con un dataset reducido con un modelo igual que el nuestro pero añadiendo un mecanismo de atención al decoder (figura 4.6). El mecanismo de atención permite que el decoder ponga más peso (atención) a ciertas partes de la entrada, las salidas del encoder en este caso, en vez de que todas ellas valgan lo mismo.

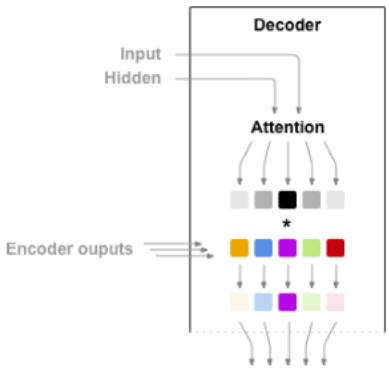


Figura 4.6: Estructura del modelo con el mecanismo de atención

Las pruebas sobre este modelo se llevan a cabo con 1000 noticias para el set de entrenamiento y una longitud de 100 palabras. Dado que el número de noticias es inferior y la longitud de estas también lo es, el vocabulario queda reducido también a 12111 palabras que para la prueba codificamos usando vectores one hot.
Con estas noticias, obtenemos que la pérdida media sobre el conjunto de entrenamiento (figura 4.7) es pequeña, rondando $0{,}0660$, no obstante, el modelo sobre-entrena y le cuesta generalizar.

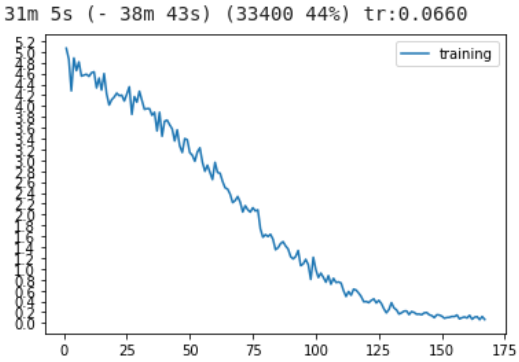


Figura 4.7: Gráfica del entrenamiento del modelo con atención

Se muestran algunos de estos resultados en los cuadros 4.9 y 4.10 pudiendo ver que las predicciones sobre el conjunto de training son favorables y el modelo se beneficia del mecanismo de atención añadido, y, en los cuadros 4.11 y 4.12 las predicciones sobre el conjunto de test.

Noticia: *la deteccion de toxinas en el aire y la incertidumbre que rodea al vertedero de zaldivar hacen crecer la alarma entre las cerca de . personas que viven en los municipios de sus alrededores . sociedad*
Titular real: *dos semanas bajo el acecho del vertedero de zaldivar*
Titular predicho: *dos semanas bajo el acecho del vertedero de zaldivar end*
Loss: 0,038022422790527345

Cuadro 4.9

Noticia: *la pretension del ministerio del interior que dirige fernando grande marlaska de endurecer lascondicionesparaelderechode asilo causo ayer el primer choque enelgobiernodecoalicion. podemos impugno el borrador de trabajo de interior. politica p.*
Titular real: *la politica migratoria causa el primer choque dentro del gobierno*
Titular predicho: *la politica migratoria causa el primer choque dentro gobierno del gobierno end*
Loss: 0,9569022872231223

Cuadro 4.10

Noticia: *la coruna acoge entre hoy y el domingo el primer torneo del kia padel fest un circuito pensado para los millones de aficionados. el resto de ciudades seran gijon de marzo madrid de abril sevilla de mayo zaragoza de junio malaga de julio alicante de septiemb re y barcelona de octubre.*
Titular real: *el primer torneo del kia padel fest comienza en la coruna*
Titular predicho: *duelo duelo moviliza humillacion apodera apodera apodera espanoles apodera apodera espanoles manos manos exhumacion exhumacion nevada mwc nevada puede nevada puede nevada puede nevada elecciones cafeteria italiana nevada italiana temerosos estados planta davis nevada aguanta intu intu reparto reparto reparto reparto reparto intu intu reparto reparto reparto reparto intu intu reparto reparto reparto reparto reparto reparto reparto partido randstad friccion randstad rafael intu intu intu noticias rafael reparto intu intu intu reparto reparto reparto reparto intu intu reparto reparto reparto reparto reparto reparto reparto reparto reparto reparto intu intu reparto reparto reparto intu intu intu reparto reparto reparto reparto*
Loss: 8,116441210110983

Cuadro 4.11

Noticia:*ha fallecido cristianamente en barcelona a la edad de anos el dia de febrero del . e .p .d . su esposa hijos nieta y demas familia lo comunican a sus amigos y alumnos y les ruegan un recuerdo en sus oraciones. la ceremonia tendra lugar manana dia de febrero del a las. horas en el tanatori sancho de avila.*
Titular real: *paulino oviedo ubierna el profe*
Titular predicho: *principe intu intu reparto reparto reparto reparto reparto partido nevada randstad catalana aislados antenas aclare aclare global global emplaza exconsellers asilo talibanes nevada friccion rafael antenas aclare global global emplaza asilo nevada exconsellers intu intu reparto reparto reparto reparto reparto reparto intu intu reparto reparto reparto reparto reparto reparto reparto reparto reparto reparto intu intu reparto reparto reparto reparto reparto reparto reparto reparto reparto reparto reparto reparto intu intu reparto reparto reparto reparto reparto reparto reparto reparto intu reparto reparto reparto reparto reparto reparto reparto intu reparto reparto reparto reparto intu reparto reparto reparto reparto reparto reparto reparto reparto reparto*
Loss: 8,083713463374547

Cuadro 4.12

En conclusión, aunque hemos obtenido resultados prometedores con el entrenamiento de datasets muy reducidos y vocabularios pequeños, la tarea de la generación de titulares no deja de ser un problema altamente complejo para el que se requiere una gran cantidad de tiempo de entrenamiento con un dataset de nuestro tamaño y que demuestra ser complicado de generalizar.

5

Conclusiones

La motivación que impulso este Trabajo de Fin de Grado era la de aprender sobre técnicas recientes de sumarización automática y replicar los resultados de dichas técnicas, normalmente utilizados en datasets en inglés, en noticias en español.

El trabajo a lo largo del último año ha sido extenso: he estudiado las distintas técnicas empleadas para la sumarización extractiva; analizado las técnicas deep learning aplicadas al Procesamiento del Lenguaje Natural, las redes LSTM y las arquitecturas Seq2Seq; he buscado datasets que me permitieran llevar a cabo las dos partes del proyecto propuesto y he estudiado los diferentes tipos de embeddings existentes.

Dentro del campo de la sumarización extractiva, basándonos en el estudio del estado del arte llevado a cabo, hemos obtenido resultados coherentes implementando dos algoritmos y analizado las ventajas que ofrecía uno sobre el otro así como los problemas que hemos encontrado comunes.

En cuanto a la sumarización abstractiva, esta tarea me ha obligado a estudiar y aprender sobre arquitecturas y redes que me eran completamente desconocidos. El desarrollo del modelo y su entrenamiento ha necesitado de una gran cantidad de esfuerzo y tiempo para el ajuste y corrección de este.

En la práctica, tras todo el trabajo llevado a cabo, para la sumarización extractiva emplearía métodos similares al algoritmo 3.1 pues ha resultado producir buenos sumarios sin los problemas que ocasiona a nivel de memoria y coste la generación de las matrices de similitud del segundo. Además, es claro que los métodos extractivos de sumarización son más prácticas ya que no necesitan de un entrenamiento previo. En cuanto a sumarización abstractiva, usaría métodos de atención para ayudar en el aprendizaje a la red ya que no solo los resultados en training han sido mejores sino que el entrenamiento ha sido más rápido.

En definitiva, la realización de este proyecto me ha permitido adquirir conocimientos nuevos sobre diversos métodos y técnicas empleadas dentro del campo del PLN e implementarlos para obtener una visión más general sobre los problemas y ventajas que traen cada uno, y enfrentarme a tareas complejas que son de gran relevancia en la actualidad.

5.1. Trabajo futuro

El trabajo futuro a llevar a cabo se extiende en varias direcciones.

Por un lado, dado que la tarea de generación de titulares no ha dado los resultados esperados, convendría el análisis de otros modelos que se hayan usado para este mismo fin y el estudio profundo del mecanismo de atención y otras técnicas para la mejora de los resultados. También la implementación del mecanismo de atención con una máquina más potente para poder entrenar un volumen mayor de noticias.

En lo que se refiere a la sumarización extractiva, estamos contentos con los resultados obtenidos. Aún así, sería interesante estudiar las aproximaciones al sumario extractivo a través de métodos deep learning y el análisis de sus resultados en comparación con los obtenidos.

Bibliografía

[1] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: an introduction," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 544–551, 2011.

[2] T. E. Doszkocs, "Natural language processing in information retrieval," *Journal of the American Society for Information Science*, vol. 37, no. 4, pp. 191–196, 1986.

[3] M. M. Lopez and J. Kalita, "Deep learning applied to nlp," *arXiv preprint arXiv:1703.03091*, 2017.

[4] F. Chollet, *Deep Learning with Python*. Manning, Nov. 2017.

[5] J. Nivre, "On statistical methods in natural language processing," *Promote IT Second Conference for the Promotion of Research in IT at New Universities and University Colleges in Sweden*, 01 2002.

[6] M. Vallez and R. Pedraza-Jimenez, "El procesamiento del lenguaje natural en la recuperaci," 2007.

[7] J. Moreno, *Automatic text summarization /*. 2014.

[8] C.-Y. Lin and E. Hovy, "From single to multi-document summarization," in *Proceedings of the 40th annual meeting of the association for computational linguistics*, pp. 457–464, 2002.

[9] K. Kaikhah, "Automatic text summarization with neural networks," in *2004 2nd International IEEE Conference on'Intelligent Systems'. Proceedings (IEEE Cat. No. 04EX791)*, vol. 1, pp. 40–44, IEEE, 2004.

[10] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut, "Text summarization techniques: a brief survey," *arXiv preprint arXiv:1707.02268*, 2017.

[11] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research and Development*, vol. 2, no. 2, pp. 159–165, 1958.

[12] L. Vanderwende, H. Suzuki, C. Brockett, and A. Nenkova, "Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion," *Inf. Process. Manage.*, vol. 43, p. 1606–1618, Nov. 2007.

[13] A. Nenkova and L. Vanderwende, "The impact of frequency on summarization," *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005*, vol. 101, 2005.

[14] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, p. 513–523, Aug. 1988.

[15] M. Franceschet, "Pagerank: Standing on the shoulders of giants," *Commun. ACM*, vol. 54, p. 92–101, June 2011.

[16] R. Mihalcea and P. Tarau, "TextRank: Bringing order into text," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, (Barcelona, Spain), pp. 404–411, Association for Computational Linguistics, July 2004.

[17] T. Schnabel, I. Labutov, D. Mimno, and T. Joachims, “Evaluation methods for unsupervised word embeddings,” in *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 298–307, 2015.

[18] D. Smilkov, N. Thorat, and C. Nicholson, “Embedding projector - visualization of high-dimensional data.”

[19] R. Kulshrestha, “Nlp 101: Word2vec-skip-gram and cbow,” May 2020.

[20] T. Mikolov, Q. V. Le, and I. Sutskever, “Exploiting similarities among languages for machine translation,” *arXiv preprint arXiv:1309.4168*, 2013.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[22] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[23] T. Ganegedara, “Light on math ml: Intuitive guide to understanding glove embeddings,” May 2019.

[24] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[25] E. Reiter and R. Dale, “Building applied natural language generation systems,” *Natural Language Engineering*, vol. 3, no. 1, pp. 57–87, 1997.

[26] E. Goldberg, N. Driedger, and R. I. Kittredge, “Using natural-language processing to produce weather forecasts,” *IEEE Expert*, vol. 9, no. 2, pp. 45–53, 1994.

[27] K. Binsted, A. Cawsey, and R. Jones, “Generating personalised patient information using the medical record,” in *Conference on Artificial Intelligence in Medicine in Europe*, pp. 29–41, Springer, 1995.

[28] A. Gatt and E. Krahmer, “Survey of the state of the art in natural language generation: Core tasks, applications and evaluation,” *Journal of Artificial Intelligence Research*, vol. 61, pp. 65–170, 2018.

[29] C. E. Shannon, “A mathematical theory of communication,” *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[30] Sciforce, “A comprehensive guide to natural language generation,” Jul 2019.

[31] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, “Recurrent neural network based language model,” in *Eleventh annual conference of the international speech communication association*, 2010.

[32] M. Sundermeyer, R. Schlüter, and H. Ney, “Lstm neural networks for language modeling,” in *Thirteenth annual conference of the international speech communication association*, 2012.

[33] A. LeNail, “Nn-svg: Publication-ready neural network architecture schematics,” *Journal of Open Source Software*, vol. 4, no. 33, p. 747, 2019.

[34] P. Sharma, “Introduction to neural networks: Deep learning,” May 2020.

[35] J. Nabi, “Recurrent neural networks (rnns),” Jul 2019.

[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[37] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *arXiv preprint arXiv:1808.03314*, 2018.

[38] R. Nallapati and Zhou, "Abstractive text summarization using sequence-to-sequence RNNs and beyond," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, (Berlin, Germany), pp. 280–290, Association for Computational Linguistics, Aug. 2016.

[39] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[40] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, pp. 1310–1318, 2013.

[41] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[42] M. Chablani, "Sequence to sequence model: Introduction and concepts," Jun 2017.

[43] L. Liu, Y. Lu, M. Yang, Q. Qu, J. Zhu, and H. Li, "Generative adversarial network for abstractive text summarization," 2018.

[44] J. Cheng and M. Lapata, "Neural summarization by extracting sentences and words," *ArXiv*, vol. abs/1603.07252, 2016.

[45] R. D. Lins, H. Oliveira, L. Cabral, J. Batista, B. Tenorio, R. Ferreira, R. Lima, G. de França Pereira e Silva, and S. J. Simske, "The cnn-corpus: A large textual corpus for single-document extractive summarization," in *Proceedings of the ACM Symposium on Document Engineering 2019*, DocEng '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[46] R. D. Lins, H. Oliveira, L. Cabral, J. Batista, B. Tenorio, D. A. Salcedo, R. Ferreira, R. Lima, G. de França Pereira e Silva, and S. J. Simske, "The cnn-corpus in spanish: A large corpus for extractive text summarization in the spanish language," in *Proceedings of the ACM Symposium on Document Engineering 2019*, DocEng '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[47] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[48] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[49] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, (Barcelona, Spain), pp. 74–81, Association for Computational Linguistics, July 2004.

[50] A. Ramirez-Noriega, R. Juárez-Ramírez, S. Jimenez, S. Inzunza, and Y. Martinez-Ramirez, "Ashur: Evaluation of the relation summary-content without human reference using rouge," *Computing and Informatics*, vol. 37, no. 2, pp. 509–532, 2018.

[51] J.-M. Torres-Moreno, H. Saggion, I. d. Cunha, E. SanJuan, and P. Velázquez-Morales, "Summary evaluation with and without references," *Polibits*, no. 42, pp. 13–20, 2010.

[52] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, "FLAIR: An easy-to-use

framework for state-of-the-art NLP," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, (Minneapolis, Minnesota), pp. 54–59, Association for Computational Linguistics, June 2019.

[53] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[54] UKPLab, "Ukplab/sentence-transformers," May 2020.

Acrónimos

ASHuR Assessing Summaries without Human reference using ROUGE.

ATS Automatic Text Summarization.

BERT Bidirectional Encoder Representations from Transformers.

CBOW Continuous Bag Of Words.

FRESA Framework for Evaluating Summaries Automatically.

GLN Generación de Lenguaje Natural.

GloVe Global Vectors for Word Representation.

IR Information Retrieval.

LCS Longest Common Subsequence.

LSTM Long Short-Term Memory.

NLP Natural Language Processing.

PLN Procesamiento del Lenguaje Natural.

RAE Real Academia Española.

RN Red Neuronal.

RNN Red Neuronal Recurrente.

ROUGE Recall-Oriented Understudy for Gisting Evaluation.

TFG Trabajo de Fin de Grado.

TF-IDF Term Frequency Inverse Document Frequency.

TIC Tecnologías de la Información y la Comunicación.

Apéndices

A

Resultados sumarios extractivos

A.1. Ejemplo 1

Atlas se mantiene en la lucha por no descender tras vencer 2-3 a Santos. Los rojinegros mantienen cinco puntos de diferencia sobre el Atlante que este viernes derrotó 1-2 al Morelia. El Atlas derrotó al Santos Laguna tres goles contra dos, con dos tantos clave del chileno Rodrigo Millar, la segunda victoria del equipo jalisciense en nueve jornadas del torneo Clausura 2014 del futbol mexicano y que lo mantiene en la lucha por no descender. Los dos goles de Millar más el del brasileño Maikon Leite marcaron una gran remontada del Atlas tras verse abajo en el marcador en el nuevo estadio Corona del Santos, por un gol del colombiano Carlos Darwin Quintero y otro del mexicano Javier Abella para los locales. Atlas conquistó su segundo triunfo del torneo y con los tres puntos que se llevó de la cancha del Santos se coloca en el decimocuarto de la tabla general, mientras Santos Laguna –líder del grupo 8 de la Copa Libertadores–se queda en el décimo. En el segundo partido de la jornada, Atlante derrotó 2-1 al Morelia como visitante, con un gol de último minuto del ecuatoriano Michael Arroyo. El colombiano Aldo Leao Ramírez por el Morelia y Roberto Gutiérrez del Atlante, tenían el partido empatado a un gol por equipo. En la Liga, el Atlante llegó a doce puntos en nueve partidos y se ha colocado en el séptimo lugar de la clasificación, mientras que el Morelia bajó al decimoquinto tras el triunfo del Atlas. La jornada nueve del Torneo de Clausura 2014 seguirá este sábado con los partidos entre Veracruz y Tijuana, Cruz Azul y Toluca, Monterrey y América, así como Chiapas y León. Mientras que el domingo se enfrentarán Puebla y Querétaro, Pumas y Pachuca , así como Guadalajara y Tigres.

Cuadro A.1: Texto original de una noticia del dataset

Los dos goles de Millar más el del brasileño Maikon Leite marcaron una gran remontada del Atlas tras verse abajo en el marcador en el nuevo estadio Corona del Santos, por un gol del colombiano Carlos Darwin Quintero y otro del mexicano Javier Abella para los locales. Atlas conquistó su segundo triunfo del torneo y con los tres puntos que se llevó de la cancha del Santos se coloca en el decimocuarto de la tabla general, mientras Santos Laguna –líder del grupo 8 de la Copa Libertadores– se queda en el décimo . En el segundo partido de la jornada, Atlante derrotó 2-1 al Morelia como visitante, con un gol de último minuto del ecuatoriano Michael Arroyo. La jornada nueve del Torneo de Clausura 2014 seguirá este sábado con los partidos entre Veracruz y Tijuana, Cruz Azul y Toluca, Monterrey y América, así como Chiapas y León.

Cuadro A.2: Resultados obtenidos por el algoritmo 3.1 para la noticia con texto original mostrado en A.1

El Atlas derrotó al Santos Laguna tres goles contra dos, con dos tantos clave del chileno Rodrigo Millar, la segunda victoria del equipo jalisciense en nueve jornadas del torneo Clausura 2014 del futbol mexicano y que lo mantiene en la lucha por no descender. Atlas conquistó su segundo triunfo del torneo y con los tres puntos que se llevó de la cancha del Santos se coloca en el decimocuarto de la tabla general, mientras Santos Laguna –líder del grupo 8 de la Copa Libertadores–se queda en el décimo. Los dos goles de Millar más el del brasileño Maikon Leite marcaron una gran remontada del Atlas tras verse abajo en el marcador en el nuevo estadio Corona del Santos, por un gol del colombiano Carlos Darwin Quintero y otro del mexicano Javier Abella para los locales. En la Liga, el Atlante llegó a doce puntos en nueve partidos y se ha colocado en el séptimo lugar de la clasificación, mientras que el Morelia bajó al decimoquinto tras el triunfo del Atlas.

Cuadro A.3: Resultados obtenidos por el algoritmo 3.3 para la noticia con texto original mostrado en A.1

A.2. Ejemplo 2

México gana bronce en clavados sincronizados en Serie Mundial de Beijing. Laura Sánchez y Arantxa Chávez quedaron en tercer puesto con 296 puntos; fueron superadas por las parejas de China y Canadá. Las clavadistas Laura Sánchez y Arantxa Chávez ganaron este viernes la medalla de bronce en la prueba de 3 metros sincronizados, en el primer día de la Serie Mundial de Clavados FINA en Beijing, China. Sánchez y Chávez obtuvieron 296.10 puntos, con parciales de 48.00 (101B), 43.80 (301B), 68.40 (405B), 69.30 (205B) y 66.60 (5152B) en las cinco rondas. El oro lo ganaron las chinas Tingmao Shi y Minxia Wu, con 330.60 puntos, seguidas de Jennifer Abel y Pamela Ware, de Canadá, con 305.16, informó la Comisión Nacional de Cultura Física y Deporte (Conade). Las mexicanas ejecutaron la misma serie de clavados que en la Serie Mundial FINA 2013 en Barcelona, pero incrementaron sus puntuaciones respecto a esa ocasión cuando obtuvieron 290.70, y fueron superadas por las parejas de China, Italia y Canadá. Las ganadoras de la presea de plata en Londres 2012, Paola Espinosa y Alejandra Orozco, quedaron en la quinta posición en la prueba de saltos sincronizados desde la plataforma, con un resultado de 300.27 puntos, informó la Conade . La competencia la ganaron las chinas Roulin Chen y Huixia Liu, con 347.52, seguidas de Meaghan Benfeito y Reseline Filion, de Canadá, con 307.56, así como de las malayas Mun Yee Leong y Pandelela Pamg, con 303. Rommel Pacheco y Jahir Ocampo, medallistas de bronce en Barcelona 2013, concluyeron la participación de los mexicanos este viernes en el Cubo de Agua, en Beijing, con el sexto lugar en clavados sincronizados de 3 metros con 409.92 puntos, señaló la Conade. Los campeones de la prueba fueron Evgeny Kuznetsov e Illya Zakharov, de Rusia, con 455.28 puntos, mientras que la plata fue para los anfitriones Yuan Cao y Yue Lin, con 450. El bronce lo obtuvieron los ucranianos Oleksandr Gorshkovozav e Illya Kvasha, con 425. La participación de los mexicanos en el torneo continuará este sábado cuando Paola Espinosa y Laura Sánchez compitan en la prueba de trampolín individual.

Cuadro A.4: Texto original de una noticia del dataset

10 puntos, con parciales de 48 . 00 (101B), 43 . 80 (301B), 68 . 40 (405B), 69 .

Cuadro A.5: Resultados obtenidos por el algoritmo 3.1 para la noticia con texto original mostrado en A.4

Rommel Pacheco y Jahir Ocampo, medallistas de bronce en Barcelona 2013, concluyeron la participación de los mexicanos este viernes en el Cubo de Agua, en Beijing, con el sexto lugar en clavados sincronizados de 3 metros con 409 . 60 puntos, seguidas de Jennifer Abel y Pamela Ware, de Canadá, con 305 . (CNNMéxico) – Las clavadistas Laura Sánchez y Arantxa Chávez ganaron este viernes la medalla de bronce en la prueba de 3 metros sincronizados, en el primer día de la Serie Mundial de Clavados FINA en Beijing, China . Las mexicanas ejecutaron la misma serie de clavados que en la Serie Mundial FINA 2013 en Barcelona, pero incrementaron sus puntuaciones respecto a esa ocasión cuando obtuvieron 290 .

Cuadro A.6: Resultados obtenidos por el algoritmo 3.3 para la noticia con texto original mostrado en A.4

B

Generación de titulares

Anexo relacionados con el capítulo de Generación de titulares.

B.1. Creación de las matrices

Las caras eran largas. Los militares estaban serios. Así llegó también al Palacio Real el candidato a presidente, Pedro Sánchez, acompañado de su ministro del Interior, Fernando Grande-Marlaska –igualmente serio–, y de su titular de Defensa, Margarita Robles, quien respondía con sonrisas a los saludos de sus mandos militares. Era un día de celebración pero bien poco lo parecía. [...]

Cuadro B.1: Extracto de una noticia

['start', 'las', 'caras', 'eran', 'largas', 'end']
['start', 'los', 'militares', 'estaban', 'serios', 'end']
['start', 'así', 'llegó', 'también', 'al', 'palacio', 'real', 'el', 'candidato', 'a', 'presidente', 'pedro', 'sánchez', 'acompañado', 'de', 'su', 'ministro', 'del', 'interior', 'fernando', 'grande', 'marlaska', 'y', 'de', 'su', 'titular', 'de', 'defensa', 'margarita', 'robles', 'quien', 'respondía', 'con', 'sonrisas', 'a', 'los', 'saludos', 'de', 'sus', 'mandos', 'militares', 'end']
['start', 'era', 'un', 'día', 'de', 'celebración', 'pero', 'bien', 'poco', 'lo', 'parecía', 'end']

Cuadro B.2: Matriz creada siguiendo el algoritmo 3.4 del extracto anterior


UAM
UNIVERSIDAD AUTONOMA
DE MADRID